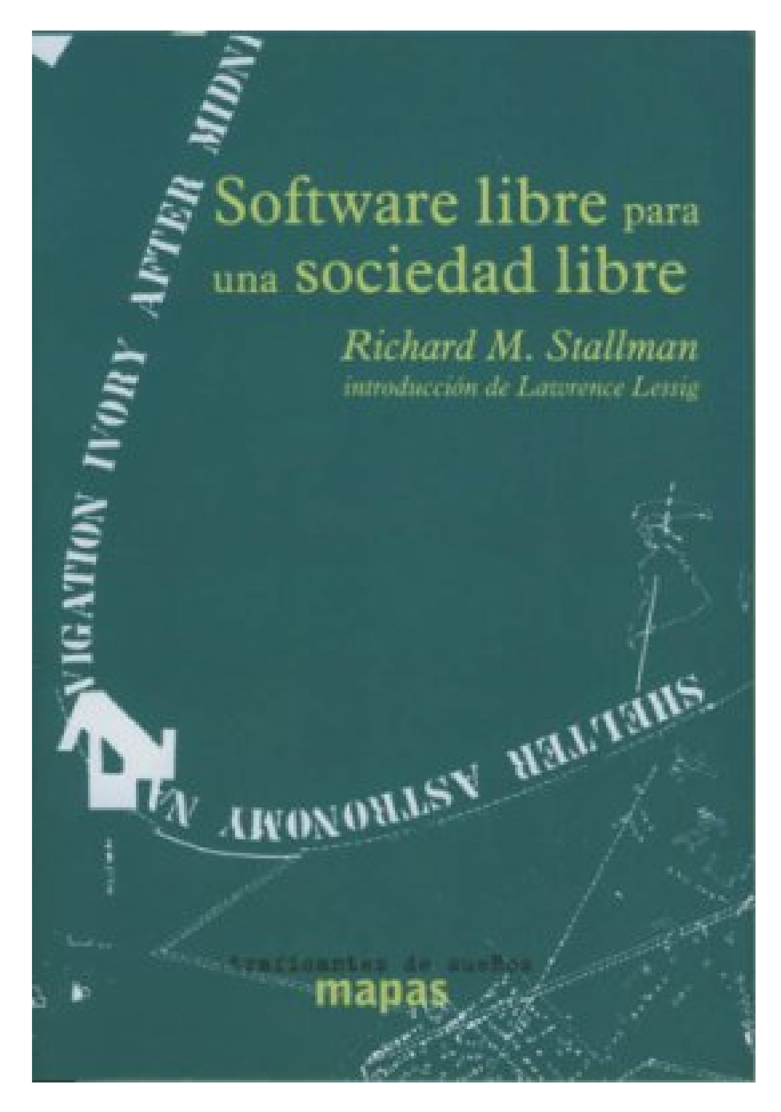
Software libre para una sociedad libre

Richard M. Stallman
Introducción de Lawrence Lessig



Diciembre 2004
Versión 1.0

Software libre para una sociedad libre
Richard M. Stallman

Título original: *Free Software, Free Society: Selected Essays of Richard M. Stallman* (GNU Press, 2002)
Primera edición en castellano (en papel): Noviembre 2004

Traducción principal: Jaron Rowan, Diego Sanz Paratcha y Laura Trinidad

Edición: Traficantes de Sueños
c/ Hortaleza 19, 1º Dcha.
28004 Madrid. Tlfno: +34 1 5320928
http://traficantes.net

ISBN: 84-933555-1-8
Depósito Legal: M-44298-2004

Edición digital a cargo de: Miquel Vidal <miquel@barrapunto.com>. Esta edición electrónica se ha realizado íntegramente con software libre, mediante el procesador LaTeX 2$_\varepsilon$, GNU Emacs y AUCTeX.

Traficantes de Sueños

Traficantes de Sueños no es una casa editorial, ni siquiera una editorial independiente que contempla la publicación de una colección variable de textos críticos. Es, por el contrario, un proyecto, en el sentido estricto de «apuesta», que se dirige a cartografíar las líneas constituyentes de otros órdenes de vida. La construcción teórica y práctica de la caja de herramientas que, con palabras propias, puede componer el ciclo de luchas de las próximas décadas.

Sin complacencias con la arcaica sacralidad de la cultura, sin concesiones para con los narcisismos del genio literario, sin lealtad alguna a los usurpadores del saber, TdS adopta sin ambagajes la libertad de acceso al conocimiento. Queda, por tanto, permitida y abierta la reproducción total o parcial de los textos publicados, en cualquier formato imaginable, salvo por explícita voluntad del autor o de la autora y sólo en el caso de las ediciones con ánimo de lucro.

Omnia sunt communia!

Índice

Acerca de la presente edición

La presente edición de *Software libre para una sociedad libre* es la primera edición castellana autorizada por Richard M. Stallman de su libro Free Software, Free Society. Un exhaustivo conjunto de ensayos y artículos que recorren la década de 1990 y los primeros años del nuevo milenio, y que conforman quizás la mejor apología escrita del software libre como dispositivo de libertad y democracia. El trabajo de edición de este libro ha sido complejo y prolongado, y ha sido posible gracias únicamente a la cooperación de una multitud de personas ligadas al mundo del software libre. De este modo, el carácter colectivo, abierto y cooperativo de la elaboración de esta edición guarda no pocas similitudes con los proyectos de desarrollo de software libre. Sin embargo, la dispersión de las colaboraciones y la enorme heterogeneidad de los estilos de traducción ha obligado a realizar una extensa labor de unificación, en la que los criterios utilizados no son necesariamente los preferidos por todos los traductores. En este sentido, hemos preferido mantener el anglicismo «copyright» frente al término jurídico de «derecho de autor», más correcto en lengua castellana, no sólo por el uso amplio y extendido del término en inglés, sino también porque todas las referencias del libro son a la legislación estadounidense. También hemos traducido «library» por biblioteca, en lugar de librería, más extendido en el lenguaje técnico de programación, pero menos correcto en términos de traducción. Por otra parte en relación a las licencias GNU de la Free Software Foundation se utiliza indistintamente tanto la traducción castellana, como Licencia Pública General [General Public License], como las siglas inglesas por las que son más corrientemente conocidas, en este caso GPL o más correctamente GNU GPL.

Debido a la enorme cantidad de recursos movilizados en la edición de esta obra, resulta inexcusable citar y agradecer la labor de Vicente Ruiz Jurado y Juan Carlos Gentile, que se encargaron de recopilar y coordinar las primeras traducciones de este volumen. También de Miquel Vidal por la orientación inicial del proyecto y desde luego, el trabajo de traducción inicial de: Leovigildo García Bobadilla (introducción); César Ballardini, Ramsés Morales, César Villanueva, Óscar Méndez Bonilla y Hugo Gayosso (cap. 1); Enrique A. Sánchez Núñez, Diego Cadogan, Pablo Ruiz Múzquiz y de nuevo Hugo Gayosso (cap. 2); equipo de traductores al español de GNU (cap. 3); Stan Bark, Carlos Rega, José Manuel Benítez Sánchez, Luis M. Arteaga y Luis Bustamante (cap. 4); Pablo Chamorro C., Steve Winston y Holman Romero (cap. 6); Steve Winston, José Manuel Benítez Sánchez, Ragnar Hojland Espinosa, Ramsés Morales, Esteban Osses Anguita y Enrique A. Sánchez Núñez (cap.8); Carlos Rega y Serena Del Bianco (cap. 11);

Conrado A. Bermúdez, Viviana Cruz, Steve Winston, Luis Miguel Arteaga y Holman Romero (cap. 14); Javier Smaldone (cap. 17); Pablo Ruiz Múzquiz, Holman Romero e Iván Martínez Cortés (cap. 17); Cristian Rovner y Luis Miguel Arteaga (cap. 19); Jesús González Barahona y Pedro de las Heras Quirós (GNU GPL); Igor Tamara, Pablo Reyes y Vladimir Tamara (GNU FDL); Rafael Palomino (GNU LGPL); y de todos aquellos que puedan reconocer parte de su trabajo en este libro, pero de los que nos ha sido imposible reunir sus nombres. Por último, es necesario reconocer la cuidada labor de traducción y corrección de los traductores principales: Jaron Rowan, Diego Sanz Paratcha y Laura Trinidad. Por deseo de R. M. Stallman parte de los fondos recaudados de la venta del libro se dedicarán a la financiación de proyectos a cargo de la Fundación del Software Libre en Europa. En concreto, se destinarán 2,5 euros por cada ejemplar vendido.

Introducción

Cada generación tiene su filósofo: un escritor o un artista que plasma la imaginación de una época. A veces estos filósofos son reconocidos como tales, pero a menudo pasan generaciones antes de que se caiga en la cuenta. Sin embargo, con reconocimiento o sin él, cada época queda marcada por la gente que expresa sus ideales, sea en el susurro de un poema o en el fragor de un movimiento político.

Nuestra generación tiene un filósofo. No es un artista, tampoco un escritor profesional. Es un programador. Richard Stallman comenzó su trabajo en los laboratorios del MIT como programador y arquitecto desarrollando software de sistemas operativos. Ha desarrollado su carrera en la vida pública como programador y arquitecto fundando un movimiento por la libertad en un mundo cada vez más definido por el «código».

El «código» es la tecnología que hace que los ordenadores funcionen. Esté inscrito en el software o grabado en el hardware, es el conjunto de instrucciones, primero escritas como palabras, que dirigen la funcionalidad de las máquinas. Estas máquinas (ordenadores) definen y controlan cada vez más nuestras vidas. Determinan cómo se conectan los teléfonos y qué aparece en el televisor. Deciden si el vídeo puede enviarse por banda ancha hasta un ordenador. Controlan la información que un ordenador remite al fabricante. Estas máquinas nos dirigen. El código dirige estas máquinas.

¿Qué control deberíamos tener sobre el código? ¿Qué comprensión? ¿Qué libertad debería haber para neutralizar el control que permite? ¿Qué poder?

Estas preguntas han sido el reto de la vida de Stallman. A través de sus trabajos y de sus palabras nos ha incitado a ser conscientes de la importancia de mantener «libre» el código. No «libre» en el sentido de que los escritores del código no reciban una remuneración, sino «libre» en el sentido de que el control, que construyen los codificadores, sea transparente para todos y en el de que cualquiera tenga derecho a tomar ese control y de modificarlo a su gusto. Esto es el «software libre», «software libre» es la respuesta a un mundo construido mediante código.

«Libre». Stallman lamenta la ambigüedad de su propio término.[1] No hay nada que lamentar. Los rompecabezas obligan a la gente a pensar y el término «libre» cumple bastante bien esta función de rompecabezas. Para los oídos estadounidenses modernos, «software libre» suena utópico, imposible. Nada, ni siquiera el almuerzo, es libre. ¿Cómo podrían ser «libres» las más importantes palabras que dirigen las máquinas más

[1] Se refiere aquí, por primera vez en este libro, a la doble acepción de la palabra inglesa *free* como libre y como gratis. [*N. del E.*]

esenciales que dirigen el mundo? ¿Cómo podría una sociedad en su sano juicio aspirar a semejante ideal?

Sin embargo, el peculiar tañido de la palabra «libre» depende de nosotros y no del propio término. «Libre» tiene diferentes significados, sólo uno de ellos se refiere a «precio». Un significado de «libre» mucho más fundamental es, dice Stallman, el del término «libertad de expresión» o quizás mejor el de la expresión «trabajo libre no forzado». No libre como gratuito, sino libre en el sentido de limitado en cuanto a su control por los otros. Software libre significa un control que es transparente y susceptible de modificación, igual que las leyes libres, o leyes de una «sociedad libre», son libres cuando hacen su control cognoscible y abierto a la modificación. La intención del «movimiento software libre» de Stallman es producir código en la medida en que pueda ser transparente y susceptible de modificación haciéndolo «libre».

El mecanismo para este fin es un instrumento extraordinariamente inteligente llamado «copyleft» que se implementa a través de una licencia llamada GPL. Usando el poder del copyright, el «software libre» no sólo asegura que permanece abierto y susceptible de modificación, sino también que otro software que incorpore y use «software libre» —y que técnicamente se convierta en «obra derivada»— debe también, a su vez, ser libre. Si uno usa y adapta un programa de software libre y distribuye públicamente esa versión adaptada, la versión distribuida debe ser tan libre como la versión de la que procede. Debe hacerse así, de lo contrario se estará infringiendo el copyright.

El «software libre», como las sociedades libres, tiene sus enemigos. Microsoft ha entablado una guerra contra la GPL, alertando a quienquiera que le escuche de que la GPL es una licencia «peligrosa». El peligro a que se refiere, sin embargo, es en gran medida ficticio. Otros plantean objeciones a la «coerción» que supone el mandato de la GPL de que las versiones modificadas sean también libres. Pero una condición no es coerción. Si no es coerción que Microsoft no permita a lo usuarios distribuir versiones modificadas de Office sin pagarle (presumiblemente) millones, entonces no es coerción que la GPL establezca que las versiones modificadas del software libre sean también libres.

También están los que califican el mensaje de Stallman de demasiado extremista. Pero no es extremista. Al contrario, en un sentido obvio el trabajo de Stallman es una simple traslación de la libertad que nuestra tradición ha inscrito en el mundo anterior al código. El «software libre» asegura que el mundo gobernado por el código es tan «libre» como nuestra tradición que construyó el mundo anterior al código.

Por ejemplo: una «sociedad libre» está regulada por leyes. Pero hay límites que cualquier sociedad libre pone a esa regulación legal: ninguna sociedad que mantenga sus leyes en secreto podría llamarse, nunca, libre. Ningún gobierno que esconda sus normas a los gobernados podría incluirse, nunca, en nuestra tradición. El Derecho gobierna. Pero sólo, precisamente, cuando lo hace a la vista. Y el Derecho sólo está a la vista cuando sus términos pueden ser conocidos por los gobernados o por los agentes de los gobernados —abogados, parlamentos.

Esta condición del Derecho va más allá del trabajo de un parlamento. Pensemos en la práctica jurídica en los tribunales norteamericanos. Los abogados son contratados

por sus clientes para defender los intereses de esos clientes. En ocasiones esos intereses son defendidos en un litigio. En el curso del litigio, los abogados redactan alegaciones. Esas alegaciones, a su vez, afectan a las decisiones judiciales. Esas decisiones determinan quien gana un caso concreto o si una determinada ley guarda conformidad con una constitución.

Todos los elementos de ese proceso son libres en el sentido a que se refiere Stallman. Las alegaciones jurídicas están disponibles para su libre uso por los demás. Las argumentaciones son transparentes —lo cual es distinto a decir que son buenas— y el razonamiento puede ser utilizado sin la autorización del abogado original. Las opiniones formuladas pueden ser citadas en alegaciones posteriores. Pueden ser copiadas e incorporadas en otra argumentación u opinión. El «código fuente» del Derecho estadounidense es deliberadamente y por principio abierto y de libre uso por cualquiera. Y así lo usan libremente los abogados, ya que el secreto de una gran argumentación es que resulte original mediante la reutilización de lo que se ha hecho antes. La fuente es libre, la creatividad y una forma de economía se cimentan sobre ella.

Esta economía del código abierto —y me refiero aquí al código legal abierto— no arruina a los abogados. Las firmas de abogados tienen incentivos suficientes para redactar buenas alegaciones incluso cuando material que crean pueda ser apropiado y utilizado por cualquier otro. El abogado es un artesano cuyo trabajo es de dominio público. Sin embargo la artesanía no es caridad. Los abogados cobran, la gente no contrata ese tipo de trabajo sin un precio. Pero esa economía progresa con trabajos posteriores que se añaden a los anteriores.

Podríamos imaginar una práctica jurídica que fuese diferente, alegaciones y argumentaciones que se mantuviesen secretas, sentencias que hiciesen pública su decisión pero no sus fundamentos. Leyes que fueran guardadas por la policía y no se hiciesen públicas para nadie más. Normativas que se aplicasen sin explicar su contenido.

Podemos imaginar esa sociedad, pero no podemos imaginarnos llamarla «libre». Estén, o no, mejor o más eficientemente gestionados los incentivos en esa sociedad, esta no podría ser considerada libre. Los ideales de libertad, de vida en una sociedad libre, exigen algo más que una gestión eficiente. En cambio, el aperturismo y la transparencia son los límites en los cuales se construye un sistema legal, sin que se añadan nuevas ideas a conveniencia de los líderes. La vida sometida al código informático no debería ser menos.

Escribir códigos no es pleitear. Es mejor, más rico, más productivo. Pero el Derecho es un ejemplo obvio de que la creatividad y la motivación no dependen de un perfecto control sobres los productos que se crean. Igual que el jazz, o las novelas, o la arquitectura, el Derecho se construye sobre el trabajo hecho con anterioridad. La creatividad siempre es esta agregación y cambio. Y una sociedad libre es aquella que garantiza que sus recursos más importantes permanecen libres, precisamente en este sentido.

Por primera vez este libro recoge los artículos y las conferencias de Richard Stallman de forma que queden claros su sutileza y su fuerza. Los ensayos abarcan un amplio espectro, desde el copyright a la historia del movimiento del software libre. Incluyen muchas argumentaciones no muy bien conocidas y, entre ellas, una apreciación espe-

cialmente inteligente sobre las cambiantes circunstancias que vuelven sospechoso al copyright en el mundo digital. Servirán como recurso para aquellos que busquen comprender el pensamiento de este hombre poderoso, poderoso por sus ideas, su pasión y su integridad, a pesar de carecer de poder en los demás sentidos. Inspirarán a aquellos que adopten estas ideas y construyan a partir de ellas.

No conozco bien a Stallman. Lo conozco lo suficientemente bien para saber que es una persona que es difícil que nos guste. Es obstinado, a menudo impaciente. Su ira puede inflamarse ante un amigo con tanta facilidad como ante un enemigo. Es testarudo y persistente, paciente en todo caso.

Pero cuando nuestro mundo finalmente comprenda el poder y el peligro del código, cuando finalmente vea que el código, como las leyes o como el gobierno, debe ser transparente para ser libre, entonces volveremos la mirada a este programador testarudo y persistente y reconoceremos la idea por cuya realidad ha luchado: la idea de un mundo donde la libertad y el conocimiento sobreviven al compilador. Y comprenderemos que nadie, por medio de sus actos o de sus palabras, ha hecho tanto para hacer posible la libertad que la sociedad venidera podría tener.

Aún no hemos ganado esa libertad. Podríamos fracasar en su consecución. Pero triunfemos o fracasemos, en estos artículos se refleja lo que esa libertad podría ser. Y en la vida que plasman esas palabras y obras está la inspiración para todo el que, como Stallman, lucha para crear esa libertad.

LAWRENCE LESSIG
Presidente de Creative Commons

Parte I

El proyecto GNU y el software libre

Capítulo 1
El Proyecto GNU[1]

La primera comunidad que comparte software

Cuando entré a trabajar en el Laboratorio de Inteligencia Artificial (AI Lab) del MIT en 1971, pasé a formar parte de una comunidad que compartía software y llevaba haciéndolo durante años. El acto de compartir software no se circunscribe a nuestra comunidad en particular: es tan antiguo como los propios ordenadores, lo mismo que compartir recetas es tan viejo como la cocina. Simplemente, nosotros lo hacíamos en mayor medida.

En el AI Lab se utilizaba un sistema operativo de tiempo compartido llamado ITS (*Incompatible Timesharing System*), diseñado y escrito por los hackers de la plantilla del lab en lenguaje ensamblador para el Digital PDP-10, uno de los ordenadores más grandes de la época. Como miembro de esta comunidad y hacker de sistemas para el AI Lab, mi labor consistía en mejorar dicho sistema.

No llamábamos «software libre» a nuestro software porque el término no existía todavía; pero era exactamente eso. Cuando alguien de otra universidad o de otra empresa quería instalar y utilizar un programa, se lo prestábamos de buen grado. Si descubrías a alguien utilizando un programa poco habitual e interesante, siempre podías preguntarle por el código fuente, leerlo, modificarlo o canibalizar partes de él para montar un programa nuevo.

El uso de la palabra «hacker» para definir al «que rompe sistemas de seguridad» es una confusión promovida por los medios de masas. Nosotros, los hackers, nos negamos a reconocer esta acepción y seguimos utilizando este término para describir a «alguien que ama la programación y disfruta explorando nuevas posibilidades».[2]

[1]Publicado originalmente en el libro colectivo *Open Sources: Voice from the Open Source Revolution*, O'Reilly, 1999

[2]Resulta difícil dar con una definición sencilla de algo tan variado como es el *hacking*, pero creo que lo que la mayor parte de los hackers tienen en común es la pasión lúdica, la inteligencia y la voluntad de exploración. Podemos decir que el *hacking* significa explorar los límites de lo posible con un espíritu de sagacidad imaginativa. Cualquier actividad en la que se despliegue esta sagacidad tiene «valor» para el hacker. Puedes ayudar a subsanar este malentendido haciendo una simple distinción entre la intromisión en la seguridad de un sistema y las actividades de *hacking*, empleando el término *cracking* para la primera. Quienes se dedican a esto se denominan *crackers*. Es posible que un cracker sea también hacker, o ajedrecista, o golfista; pero la mayoría no lo son («On Hacking», RMS; 2002).

EL colapso de la comunidad

La situación cambió drásticamente a principios de los años ochenta, con la desaparición de la comunidad hacker del AI Lab, seguida de la desaparición del ordenador PDP-10.

En 1981, la empresa pionera Symbolics contrató a casi todos los hackers del AI Lab, y nuestra diezmada comunidad fue incapaz de sobrevivir. (En el libro *Hackers*, Stephen Levy describe estos acontecimientos, a la vez que nos proporciona un panorama bastante preciso de lo que fue la época dorada de esta comunidad). Cuando el AI Lab compró un nuevo PDP 10 en 1982, sus administradores decidieron usar un sistema de Digital de tiempo compartido no libre en lugar del ITS en la nueva máquina.

Poco después, Digital dejó de fabricar la serie PDP-10. Su arquitectura elegante y poderosa de los años sesenta no podía adaptarse de forma natural a los grandes espacios de direccionamiento característicos de los años ochenta. Lo cual explica que casi todos los programas que integraban el sistema ITS resultaran obsoletos. De esa manera se enterraba definitivamente al ITS: quince años de trabajo tirados por la borda.

Los modernos ordenadores de la época, como el VAX o el 68020, contaban con su propio sistema operativo, pero ninguno utilizaba software libre. Había que firmar un acuerdo de confidencialidad incluso para obtener una copia ejecutable.

Todo ello significaba que antes de poder utilizar un ordenador tenías que prometer no ayudar a tu vecino. Quedaban así prohibidas las comunidades cooperativas. Los titulares de software propietario establecieron la siguiente norma: «Si compartes con tu vecino, te conviertes en un pirata. Si quieres hacer algún cambio, tendrás que rogárnoslo».

La idea de que el sistema social en torno al software propietario —un sistema que te impide compartir o modificar el software— es antisocial, poco ético, sencillamente equivocado, puede sorprender a algunos lectores. Pero ¿qué podemos decir acerca de un sistema que siembra la división entre el público y abandona a los usuarios a la indefensión más absoluta? Estos lectores probablemente hayan asumido el sistema social asociado con el software propietario como algo inevitable o habrán considerado la cuestión de la misma forma que se plantea por parte de las empresas de software propietario. Los editores de software se han esforzado mucho en convencernos de que sólo hay una forma de abordar esta cuestión.

Cuando los editores de software hablan de «ejercer» sus «derechos» o de «acabar con la piratería», lo que dicen es, de hecho, secundario. El verdadero mensaje de estas declaraciones se esconde en ciertas presunciones implícitas que dan por supuestas; creen que el público debe aceptarlas sin cuestionarlas. De modo que analicémoslas.

Una suposición es que las empresas de software tienen el derecho natural e incuestionable a poseer software, y por ende a detentar todo el poder sobre sus usuarios. (Si de verdad se tratara de un derecho natural, nosotros no objetaríamos nada, independientemente del perjuicio que esto ocasionara al público.) Pero lo interesante es que la Constitución de los EE.UU. y el derecho tradicional rechazan este punto de vista. El

copyright no es una ley natural, sino un monopolio artificial impuesto por el Estado que limita el derecho natural de los usuarios a copiar.

Otra presunción implícita es que lo único importante en el software es la función que te permite desempeñar —que, como usuarios de ordenadores, no deberíamos preocuparnos de que tipo de sociedad se nos permite tener.

Una tercera presunción es que no dispondríamos de software de utilidad —o de un programa para realizar esta u otra tarea— si no cedemos el derecho de los usuarios sobre un programa a la empresa responsable del mismo. Esto resultaba convincente antes de que el movimiento del software libre demostrara que podíamos crear muchísimos programas, y muy útiles, sin necesidad de cadenas.

Si preferimos rechazar estas presunciones y analizamos estas cuestiones de acuerdo con los criterios morales y el sentido común del ciudadano de a pie, anteponiendo a los usuarios a cualquier otra consideración, llegaremos a conclusiones muy diferentes. Los usuarios de ordenadores deberían ser libres para modificar los programas y ajustarlos a sus necesidades, libres para compartirlos, porque la cooperación con los demás constituye la base de la sociedad.

Una elección moral radical

Una vez desapareció mi comunidad, era imposible seguir como hasta entonces. De modo que me enfrenté a un dilema moral radical.

Lo más fácil hubiera sido subirme al tren del software propietario, firmar acuerdos de confidencialidad y prometer no ayudar a mis compañeros hackers. Es muy probable que ahora me dedicara a desarrollar software publicado con cláusulas de confidencialidad, presionando así a otros para traicionar también a sus compañeros.

Podría haber ganado mucho dinero de esta forma, y quizás me hubiera divertido escribiendo código. Pero sabía que, al final de mi carrera, echaría la vista atrás y sólo habría contribuido a levantar muros para dividir a la gente, habría pasado toda mi vida convirtiendo este mundo en un lugar mucho peor.

Ya había experimentado lo que se siente al firmar un acuerdo de confidencialidad cuando una persona se negó a entregarnos, a mí y al AI Lab, el código fuente del programa de control de nuestra impresora. (La ausencia de ciertas funciones en este programa convertía el uso de la impresora en una experiencia muy frustrante.) De modo que no podía engañarme sobre la inocencia de estos acuerdos. Monté en cólera cuando aquel individuo se negó a compartirlo con nosotros. No podía hacerle lo mismo al resto del mundo.

Otra opción, más directa aunque desagradable, hubiera sido abandonar el mundo de los ordenadores. De esa manera no malgastaría mis aptitudes, aunque con todo seguirían sin servir de nada. No sería culpable de dividir y restringir libertad a los usuarios de ordenadores, pero eso llegaría tarde o temprano.

Decidí estudiar la manera en que un programador podría hacer algo por el bien común. Me pregunté si podía escribir uno o varios programas que permitiesen resucitar nuevamente a nuestra extinta comunidad.

La respuesta era obvia: la primera cosa necesaria era crear un sistema operativo, el software crucial para empezar a utilizar un ordenador. Con un sistema operativo puedes hacer muchas cosas; sin él, ni siquiera puedes hacer funcionar un ordenador. Mediante un sistema operativo libre podríamos armar una nueva comunidad cooperativa de hackers —e invitar a todos a que se uniesen a ella. Y cualquiera podría utilizar un ordenador sin verse obligado previamente a conspirar para privar de esto a sus amigos.

Como desarrollador de un sistema operativo, tenía las aptitudes necesarias para desempeñar esta labor. De manera que, aun cuando el éxito no estuviera asegurado, comprendí que había sido elegido para llevar a cabo esta misión. Opté por crear un sistema compatible con Unix para dotarle así de portabilidad y facilitar el cambio a los usuarios de Unix. El nombre de GNU fue elegido según una tradición de los hackers, como un acrónimo recursivo de «GNU's Not Unix».[3]

Un sistema operativo no significa sólo un kernel, que apenas permite ejecutar otros programas. En los años setenta, cualquier sistema operativo decente incluía sus propios procesadores de comandos, ensambladores, compiladores, interpretes, depuradores, editores de textos, gestores de correo y mucho más. ITS, Multics, VMS y Unix, todos incluían estos componentes.

Más adelante, escuché estas palabras, atribuidas a Hillel: «Si no actúo en mi nombre, ¿quién lo hará por mí? Y entonces, ¿en qué me convertiré? Y si ahora no, entonces ¿cuándo?».

La decisión de emprender el proyecto GNU se basaba en un espíritu similar.

Como ateo, no sigo el ejemplo de ningún líder religioso, pero a veces admiro las cosas que han llegado a decir.

Libre en su acepción de libertad

A veces se malinterpreta el término de «software libre» —para empezar, no tiene ninguna relación con el precio. Lo que nos interesa es la libertad. He aquí la definición de software libre. Un programa es software libre para el usuario siempre que, como usuario particular, tengas:

1. La libertad de ejecutar el programa sea cual sea el propósito.

2. La libertad para modificar el programa para ajustarlo a tus necesidades. (Para que se trate de una libertad efectiva en la práctica, deberás tener acceso al código fuente, dado que sin él la tarea de incorporar cambios en un programa es extremadamente difícil.)

3. La libertad de redistribuir copias, ya sea de forma gratuita, ya sea a cambio del pago de un precio.

4. La libertad de distribuir versiones modificadas del programa, de tal forma que la comunidad pueda aprovechar las mejora introducidas.

[3] En castellano, «GNU No es Unix».

Dado que nos referimos a la libertad y no al precio, no existe contradicción alguna entre la venta de copias y el software libre. De hecho, la libertad para vender copias es crucial: las colecciones de software libre a la venta en formato de CD-ROM son muy importantes para la comunidad y venderlas es una forma de recaudar fondos para el desarrollo de software libre. Por lo tanto, cualquier programa que no podamos incluir en estas colecciones no podrá calificarse de software libre.

Dada la ambigüedad del calificativo «libre», llevamos mucho tiempo buscando alternativas, pero nadie ha encontrado ninguna satisfactoria. La lengua inglesa es de las más rica en lo que a palabras y matices se refiere, pero carece de un término simple e inequívoco para «libre» en el sentido de libertad —«unfettered» [sin cadenas] sería el calificativo que más se ajusta al significado. Alternativas como «liberado», «libertad» o «abierto» no significan lo mismo o presentan otros inconvenientes.

El software GNU y el Sistema GNU

El desarrollo de un sistema operativo de principio a fin es un proyecto colosal. Como primera medida, decidí adaptar y utilizar algunas piezas existentes de software libre siempre que me fuera posible. Desde el inicio, decidí usar TEXcomo principal procesador de texto, y unos años más tarde me pasé al X Window System en vez de escribir otro sistema de ventanas para GNU.

Debido a esta decisión, el sistema GNU no consiste en una colección completa de software GNU. El sistema incluye programas desarrollados por otros individuos y para proyectos con sus propios propósitos que empleamos por su condición de software libre.

Los inicios del proyecto

En enero de 1984 abandoné mi empleo en el MIT y comencé a escribir software GNU. Abandonar el MIT era imprescindible si quería que nadie interfiriera en la distribución de GNU como software libre. De haberme quedado, el MIT podría haberse apropiado de mi trabajo e impuesto sus propios términos de distribución, o incluso convertir el trabajo en un paquete de software propietario. No tenía ninguna intención de hacer una gran cantidad de trabajo para ver como se convertía en algo inútil en relación a su propósito inicial: crear una nueva comunidad dedicada a compartir software.

No obstante, el profesor Winston, el entonces director del Lab AI en el MIT, me invitó a utilizar las instalaciones del laboratorio.

Los primeros pasos

Poco después de comenzar el proyecto GNU, me hablaron del Free University Compiler Kit, también conocido como VUCK. [La palabra danesa para libre (free) estaba escrita con una «V»] Se trataba de un compilador diseñado para trabajar con múltiples

lenguajes, incluido C y Pascal, y compatible con ordenadores de objetivos múltiples. Me puse en contacto con el autor para pedirle permiso y utilizarlo en GNU.

Me contestó burlonamente, diciendo que la universidad era gratuita, pero no el compilador. Así que decidí que el primer programa para el proyecto GNU sería un compilador capaz de trabajar en múltiples lenguajes y plataformas.

Para evitar tener que reescribir todo el compilador, obtuve el código fuente para el compilador Pastel, un compilador de plataformas múltiples desarrollado en Lawrence Livermore Lab. Soportaba, y estaba escrito, en una versión ampliada de Pascal, diseñada como lenguaje de programación de sistemas. Le añadí un *front end* C y comencé a pasarlo a un ordenador Motorola 68000, pero tuve que abandonar el intento al descubrir que el compilador requería muchos megabytes de espacio, y el sistema Unix 68000 de entonces sólo tenía capacidad para 64K.

Me di cuenta de que el compilador Pastel analizaba el archivo de entrada en forma de árbol sintáctico, convirtiéndolo en una cadena de «instrucciones» y luego generando todo el archivo de salida sin liberar espacio de almacenamiento. Así que concluí que tendría que escribir un nuevo compilador partiendo de cero. El resultado es el compilador conocido como GCC; aunque no contiene ningún elemento del compilador Pastel, conseguí adaptar y utilizar el *front end* C que había escrito. Pero eso fue años más tarde. Antes trabajé en el GNU Emacs.

GNU Emacs

Comencé a trabajar en el GNU Emacs en septiembre de 1984, y a principios de 1985 ya podía ser utilizado. Esto me permitió comenzar a usar el sistema Unix para labores de edición. Dado que nunca me interesó aprender a usar vi o ed, hasta entonces había realizado mis ediciones en otro tipo de máquinas.

En aquel momento había gente interesada en utilizar GNU Emacs, lo que planteó el problema de la distribución. Por supuesto, lo coloqué en el servidor anónimo ftp del ordenador del MIT [Este ordenador, prep.ai.mit.edu, se convirtió así en el principal sitio ftp de distribución de GNU; al desmantelarlo años más tarde, transferimos el nombre a nuestro nuevo servidor ftp]. Pero en aquel entonces, muchos no tenían acceso a Internet y no podían descargar una copia vía FTP. ¿Qué podía decirles?

Podría haberles dicho: «Busca un amigo en la red y que te haga una copia». O podría haber hecho lo mismo que hiciera con el PDP-10 Emacs original, a saber, decirles: «Envíame una cinta y un SASE, y te lo devolveré por correo con una copia de Emacs». Pero como no tenía trabajo y andaba buscando la manera de ganar dinero con el software libre, anuncié que enviaría copias a cualquiera interesado a cambio de ciento cincuenta dólares. Así comenzó mi empresa de distribución de software libre, precursora de las empresas que hoy distribuyen sistemas Linux basados en GNU.

¿Un programa es libre para cualquier usuario?

Cuando un programa de software libre deja de estar en manos de su autor, esto no significa necesariamente que siga siendo software libre para cualquiera que se haga con una copia de él. Por ejemplo, el software de dominio público —software sin *copyright*— es software libre, pero cualquiera puede modificarlo y hacer una versión propietaria a partir de él. Lo mismo ocurre con muchos programas libres con copyright que se distribuyen con licencias simples muy permisivas que autorizan el desarrollo de versiones propietarias modificadas.

El ejemplo paradigmático de este problema es el X Window System. Desarrollado en el MIT y publicado como software libre con una licencia permisiva, pronto fue adoptado por diversas empresas informáticas. Añadieron X, sólo en forma binaria, a sus sistemas propietarios Unix, siempre acompañados del clásico acuerdo de confidencialidad. Estas copias de X dejaron de ser software libre, igual que Unix.

Los desarrolladores del X Window System no lo consideraron un problema—lo esperaban y pretendían que eso ocurriera. Su objetivo no era la libertad sino el «éxito», definido en función del número de usuarios. No les importaba si éstos eran libres o no, bastaba con que fueran muchos.

Esto condujo a una situación paradójica, en la que dos maneras de medir el grado de libertad dieron respuestas distintas a la pregunta «¿Es libre este programa?» Si atendemos a la libertad que proporcionaban los términos de distribución del MIT, entonces la conclusión es que el X era software libre. Pero si tenemos en cuenta la libertad del usuario medio de X, la respuesta es que se trataba de software propietario. La mayoría de los usuarios de X utilizaban las versiones propietarias que venían con el sistema Unix, no la versión libre.

El Copyleft y la GNU GPL

El objetivo de GNU era proporcionar libertad a los usuarios, no simplemente ser popular. De modo que necesitábamos idear unos términos de distribución que impidieran que el software de GNU se convirtiera en software propietario. El método que empleamos se denominó *copyleft*.

Copyleft utiliza la ley de *copyright*, pero dándole la vuelta para servir a un propósito opuesto al habitual: en lugar de privatizar el software, ayuda a preservarlo como software libre.

La idea fundamental del copyleft es que se autoriza la ejecución del programa, su copia, modificación y distribución de versiones modificadas, siempre que no se añada ninguna clase de restricción a posteriori. De este modo, las libertades cruciales que definen el «software libre» quedan garantizadas para cualquiera que posea una copia; estas libertades se convierten en derechos inalienables.

Para que el copyleft sea efectivo las versiones modificadas deberán ser libres también. Esto garantiza que cualquier tarea basada en nuestro trabajo se pondrá a disposición de la comunidad si llegara a publicarse. Cuando los programadores que tienen

empleo se ofrecen voluntariamente a mejorar el software GNU, sólo el copyleft impide que sus jefes les digan: «No podéis compartir esos cambios, porque vamos a utilizarlos para crear nuestra versión propietaria del programa».

El requisito de que los cambios sean libres es esencial para garantizar la libertad de los usuarios del programa. Las empresas que privatizaron el X Window System incorporaron ciertos cambios para instalarlo en sus sistemas y en su hardware. Estos cambios eran pequeños comparados con la envergadura del sistema, pero no eran en absoluto triviales. Si estos cambios se esgrimían como excusa para denegar la libertad a los usuarios, cualquiera podría aprovecharse de ello.

Al combinar un programa libre con un código no libre se plantea un problema similar. Esta combinación acabaría siendo inevitablemente no libre; las libertades suprimidas en la parte no libre del programa afectarán a éste en su totalidad. Autorizar este tipo de combinaciones abriría un boquete lo bastante grande para hundir el barco entero. Por lo tanto, un objetivo crucial del copyleft es tapar este boquete: cualquier cosa añadida o combinada con un programa copyleft, para formar una versión modificada deberá preservar su condición de software libre y su copyleft.

Nosotros aplicamos una forma específica de copyleft para la mayor parte del software de GNU, conocida como la GNU General Public License o para abreviar GNU GPL. Recurrimos a otros tipos de copyleft según las circunstancias específicas. También se aplica el copyleft a los manuales de GNU, pero utilizamos una forma más sencilla, porque la complejidad de la GNU GPL resulta innecesaria en estos casos.

En 1984 o 1985, Don Hopkins —un compañero con mucha imaginación— me envió una carta. En el sobre había escrito una serie de proverbios, incluido el que sigue: «Copyleft—quedan revocados todos los derechos». Empleé la palabra «copyleft» para bautizar el concepto de distribución que andaba desarrollando en aquel momento.

La Free Software Foundation

A medida que aumentaba el interés por Emacs, otros vinieron a sumarse al proyecto GNU, y decidimos que era el momento de volver a buscar fuentes de financiación. De este modo, en 1985 creamos la Free Software Foundation, una organización sin ánimo de lucro dedicada al desarrollo de software libre. La FSF también se hizo con la empresa de distribución de copias de Emacs, a lo que más tarde añadiría otros programas *libres*— no sólo de GNU— así como la venta de manuales libres.

La FSF acepta donaciones, pero la mayor parte de sus ingresos siempre procedió de las ventas —de copias de software libre y de otros servicios relacionados con éste. En la actualidad, vende CD-Rom de códigos fuente, CD-Rom con los binarios, manuales cuidadosamente impresos —con total libertad para redistribuirlos y modificarlos— y Deluxe Distributions —colecciones enteras de software adaptadas a la plataforma de elección del cliente.

Los empleados de la Free Software Foundation han escrito y se han encargado del mantenimiento de una serie de paquetes de software de GNU. Dos ejemplos notables son la librería C y la shell. Todos los programas ejecutados en un sistema GNU/Linux

utilizan la librería C de GNU para comunicarse con Linux. Fue desarrollada por un miembro de la plantilla de la Free Software Foundation, Roland McGrath. La shell utilizada en la mayoría de los sistemas GNU/Linux se llama BASH —acrónimo de Bourne Again Shell—, desarrollada por otro empleado de la FSF, Brian Fox.

Financiamos el desarrollo de estos programas porque el proyecto GNU no se reducía exclusivamente a las herramientas o al entorno de desarrollo. Nuestra meta era un sistema operativo completo, y estos programas eran necesarios para alcanzar nuestro objetivo.

Con el nombre de «Bourne again Shell» pretendíamos mofarnos de la «Bourne Shell», la shell más común en Unix.

Los servicios relacionados con el software libre

La filosofía del software libre rechaza una práctica empresarial concreta y muy generalizada, *pero no rechaza el negocio* en general. Cuando una empresa respeta la libertad de los usuarios, le deseamos mucho éxito.

La venta de copias de Emacs ilustra una clase de empresa relacionada con el software libre. Cuando la FSF se hizo con el negocio, me vi obligado a buscarme nuevamente la vida. Así fue como empecé a vender servicios relacionados con el software libre que acababa de desarrollar. Esto incluía la enseñanza de cuestiones como la programación de GNU Emacs, la modificación del GCC a la medida del usuario o el desarrollo de software, normalmente para instalar el GCC en nuevas plataformas.

Hoy por hoy, una serie de corporaciones se dedican a este tipo de servicios relacionados con el software libre. Algunas distribuyen colecciones de software libre en CD-Rom; otras proporcionan servicio técnico a distintos niveles, contestando a las preguntas de los usuarios, subsanando *bugs* o añadiendo nuevas funciones. Incluso, estamos empezando a ver empresas dedicadas al lanzamiento de nuevos productos de software libre.

Pero debemos andarnos con cuidado —una serie de empresas asociadas con el término «código abierto» basan su mercado en el software no libre que funciona con software libre. No son empresas de software libre, su software es propietario, y con sus productos pretenden tentar a los usuarios y despojarles de su libertad. Se las conoce como empresas de «valor añadido», lo que refleja los valores que querrían que adoptásemos: la comodidad antes que la libertad. Si valoramos la libertad, deberíamos hablar de productos de «libertad sustraída».

Los objetivos técnicos

El principal objetivo de GNU era ser software libre. Aun cuando GNU no entrañara ninguna ventaja técnica frente a Unix, sí tendría una ventaja social, al permitir que los usuarios cooperaran, y otra ética, al respetar su libertad.

Pero es natural aplicar al trabajo los criterios ya conocidos de buena práctica —por ejemplo, la asignación dinámica de estructuras de datos para evitar las limitaciones de

tamaño fijadas arbitrariamente y el empleo de códigos de ocho bits, siempre que esto resultara apropiado.

Por otro lado, rechazábamos ese empeño de Unix en conservar una memoria reducida, y así decidimos no dar soporte a las máquinas de 16 bits —estaba claro que las de 32 bits serían la norma, para cuando hubiéramos terminado el sistema GNU — y no reducir la memoria a menos que superásemos un megabyte. En los programas en que no fuera crucial administrar archivos de gran tamaño, animábamos a los programadores a insertar un archivo de entrada entero en el core, luego a escanear su contenido sin preocuparse del I/O.

Estas decisiones permitieron que muchos programas GNU superasen a sus homólogos de Unix en fiabilidad y velocidad.

La donación de ordenadores

A medida que iba aumentaba la popularidad del proyecto GNU, la gente empezó a donar ordenadores que operaban con Unix. Y fueron de gran utilidad, porque la forma más fácil de desarrollar componentes de GNU era partiendo de un sistema Unix y reemplazar sus componentes uno a uno. Pero esto nos planteó un dilema ético: *¿era correcto poseer, aunque fuera tan solo una copia, de Unix?*

Unix era —y es— software propietario, y según la filosofía del proyecto GNU no debíamos recurrir a él. Pero, al aplicar la misma lógica que nos lleva a justificar el uso de la violencia en legítima defensa, concluí que era igualmente legítimo utilizar un paquete propietario cuando éste resultara crucial para desarrollar un sustituto libre que ayudaría a otros a dejar de utilizar el paquete propietario.

Pero, aun cuando los medios justificaran el fin, no dejaban de ser medios poco éticos. Hoy en día ya no tenemos ninguna copia de Unix, porque lo reemplazamos por sistemas operativos libres. Cuando no podíamos sustituir el sistema operativo de un ordenador por otro libre, entonces reemplazábamos el ordenador entero.

La lista de tareas de GNU

A medida que avanzaba el proyecto GNU y se desarrollaron o descubrieron un creciente número de componentes de sistema, nos pareció muy útil elaborar una lista de asignaturas pendientes. La utilizamos para reclutar desarrolladores que escribieran las piezas que faltaban. Esta lista se conoció como la lista de tareas de GNU. Además de los componentes de Unix, incluimos en la lista otros proyectos útiles de software y la documentación que, en nuestra opinión, precisaba cualquier sistema completo.

En la actualidad, apenas figuran algunos componentes de Unix en la lista de tareas de GNU —hemos llevado a cabo la mayor parte, a excepción de algunas menos trascendentales. Pero la lista está repleta de proyectos que podrían calificarse de «aplicaciones». Cualquier programa que despierte el interés de algo más que un reducido grupo de usuarios se añadirá al sistema operativo.

Incluso llegamos a incluir juegos en esta lista —lo hicimos desde el principio. Unix contenía juegos, así que lógicamente GNU tenía que hacer lo propio. Pero la compatibilidad nunca fue un problema para los juegos, de modo que no replicamos los de Unix. Optamos en cambio por una gama de distintas clases de juegos que pensamos podrán gustar a los usuarios.

La librería GNU GPL

La librería C GNU utiliza un copyleft especial llamado GNU Library General Public License, que autoriza el enlace de software propietario con la librería. ¿Por qué permitir esta excepción?

No es una cuestión de principios. Ningún principio establece el derecho de los productos de software propietario a incluir nuestro código —¿por qué contribuir a un proyecto que niega el derecho a compartir? El uso de la LGPL para la librería C, o para cualquier otra librería, responde más bien a una estrategia.

La librería C desempeña tareas genéricas; todo sistema o compilador propietario viene acompañado de una librería C. Por lo tanto, limitar nuestra librería C al software libre no reportaría ninguna ventaja para éste —hubiera desalentado el uso de nuestra librería.

Nuestro sistema es una excepción a este respecto: en el sistema GNU —incluido GNU/Linux—, la librería C GNU es la única en C. Por lo que los términos de distribución de la librería C GNU determinan si es posible o no compilar un programa propietario para el sistema GNU. No existen razones éticas para autorizar la incorporación de aplicaciones propietarias en el sistema GNU, pero estratégicamente parece que prohibir esto desincentivaría el uso del sistema GNU en lugar de alentar el desarrollo de aplicaciones libres.

Esta es la razón de que utilizar la Library GPL sea una buena estrategia para la librería C. Para otras librerías, la estrategia a adoptar debe estudiarse caso por caso. Si una librería desempeña una tarea especial que puede ayudar a escribir ciertos tipo de programas, publicarla con GPL, limitándola exclusivamente a los programas libres, será una manera de ayudar a otros desarrolladores de software libre, proporcionándoles una ventaja frente al software propietario.

Tomemos por ejemplo la GNU Readline,[4] una librería desarrollada para la edición de comandos para BASH. Readline se publica con una GNU GPL ordinaria, no con la Library GPL. Es indudable que esto reduce el volumen de uso de Readline, pero no supone una pérdida para nosotros. Por otro lado, se ha desarrollado al menos una aplicación útil en software libre que puede utilizar la Readline, y esto sí constituye un auténtico logro para la comunidad.

Los desarrolladores de software propietario cuentan con la ventaja que proporciona el dinero; los de software libre deben idear ventajas entre ellos. Espero que un día

[4]La librería GNU Readline provee una serie de funciones a aquellas aplicaciones que permitan a sus usuarios editar líneas de comando desde el teclado.

contemos con una amplia colección de librerías con GPL sin paralelo en el mundo del software propietario, una colección que proporcione módulos útiles que sirvan de base para el futuro software libre y entrañen una ventaja decisiva para fomentar su desarrollo.

¿Un reto personal?

Eric Raymond dice que «todo buen trabajo de software empieza cuando un desarrollador se plantea un reto personal». Es posible que esté en lo cierto, pero muchos componentes esenciales del software GNU se desarrollaron con el fin de crear un sistema operativo libre y completo. Su origen está en una visión y un plan, no en un impulso individual.

Por ejemplo, desarrollamos la librería C GNU, la Bourne Again Shell (BASH) y el GNU tar porque cualquier sistema similar a Unix precisaba de estos componentes. Lo mismo puede decirse de mis propios programas —el compilador C GNU, GNU Emacs, GDB y GNU Make.

Algunos programas GNU se desarrollaron para enfrentarse a amenazas específicas sobre nuestra libertad. Por eso desarrollamos el gzip, para sustituir al programa Compress cuando éste dejó de estar a disposición de la comunidad gracias a las patentes LZW.[5] Buscamos a gente que pudiera desarrollar el LessTif, y más recientemente GNOME y Harmony, y así abordar los problemas planteados por ciertas librerías propietarias —véase a continuación «Librerías no libres». Estamos desarrollando el GNU Privacy Guard para reemplazar el popular software de encriptación no libre, porque los usuarios no deberían verse obligados a elegir entre su privacidad y su libertad.

Claro que la gente encargada de escribir estos programas empezó a interesarse en el trabajo, y algunos añadieron muchas funciones para satisfacer sus propias necesidades e intereses. Pero esa no es la razón de la existencia de los programas.

Acontecimientos inesperados

Al iniciarse el proyecto GNU pensé que desarrollaríamos el sistema en su totalidad y luego lo publicaríamos entero. Pero no fue así.

Dado que cada uno de los componentes del sistema GNU se implantó en un sistema Unix, todos ellos podían ejecutarse en sistemas Unix mucho antes de que existiera el sistema GNU. Algunos de estos programas se hicieron muy populares y los usuarios empezaron a ampliarlos y a transportarlos —a las diversas versiones incompatibles de Unix, y también a otros sistemas.

El proceso dotó de mayor potencia a estos programas, y atrajo tanto fondos como colaboradores al proyecto GNU. Pero es probable que también retrasara la concepción de un sistema mínimamente funcional durante varios años, dado que los desarrolladores de GNU dedicaban la mayor parte de su tiempo al mantenimiento de estos puertos

[5] El algoritmo Lempel-Ziv-Welch se emplea para la compresión de datos.

y a la incorporación de funciones a los componentes existentes, en vez de escribir los que faltaban.

El GNU Hurd

En 1990, el sistema GNU estaba casi terminado. Faltaba crear un solo componente central, el kernel. Decidimos crearlo como una colección de procesos de servidor que se ejecutaría sobre Mach. Mach es un microkernel desarrollado en la Carnegie Mellon University y, más tarde, en la Universidad de Utah. El GNU Hurd es una colección de servidores —o «manada de gnus»—implantados en Mach que desempeñan las diversas tareas propias del kernel de Unix. Su desarrollo se retrasó mientras esperábamos la publicación de Mach como software libre, tal y como nos habían prometido.

Una de las razones que nos impulsó a elegir este diseño era evitar lo que parecía la parte más dura del trabajo: depurar un programa de kernel sin un depurador de fuentes. Esto ya se había resuelto en Mach, y esperábamos depurar los servidores Hurd como programas de usuarios, con el GDB. Pero pasó mucho tiempo hasta que lo logramos, y los servidores multiproceso que se envían mensajes entre sí resultan extremadamente difíciles de depurar. La consolidación del Hurd ha llevado muchísimos años.

Alix

En principio, el kernel GNU *no* iba a llamarse Hurd. Su nombre original era Alix —por mi novia de aquel momento. Ella era administradora de sistemas Unix, y advirtió que su nombre casaba perfectamente con los nombres escogidos para las distintas versiones de Unix. Bromeando, le dijo a sus amigos: «Deberían bautizar un kernel con mi nombre». No dije nada, pero decidí sorprenderla con un kernel llamado Alix.

Sin embargo, el nombre no se mantuvo. Michael Bushnell —ahora Thomas—, el principal desarrollador del kernel, prefería el nombre de Hurd, y llamó Alix a una parte del kernel —la encargada de capturar las llamadas del sistema y administrarlas enviando mensajes a los servidores Hurd.

Por fin, Alix y yo nos separamos y ella se cambió de nombre. En cualquier caso, el diseño de Hurd se modificó para que la librería C enviase mensajes directamente a los servidores, lo que supuso la desaparición del componente Alix.

Pero antes de todo esto, una amiga de Alix se encontró con el nombre en el código fuente de Hurd y se lo contó. Así que el nombre cumplió su cometido.

Linux y GNU/Linux

El GNU Hurd no está listo para producción. Afortunadamente, otro kernel estaba a nuestra disposición. En 1991, Linus Torvalds desarrolló un kernel compatible con Unix y lo llamó Linux. En el año 1992, la combinación de Linux con el incompleto sistema

GNU resultó en un sistema operativo libre. [Esta combinación fue, por supuesto, una labor extraordinaria]. Gracias a Linux podemos ejecutar hoy una versión del sistema GNU.

Denominamos esta versión GNU/Linux para explicar su composición, una combinación del sistema GNU con Linux como kernel.

Los retos futuros

Hemos demostrado ser capaces de desarrollar una amplia gama de software libre. Esto no significa que seamos invencibles e imparables. Existen diversos retos que plantean un futuro incierto para el software libre. Enfrentarnos a ellos nos exigirá un esfuerzo constante y mucha resistencia, a veces por muchos años. Necesitaremos la clase de determinación que exhibe la gente cuando valora su libertad y la protege a toda costa.

En los cuatro apartados que siguen discutiremos estos retos.

Hardware secreto

Los fabricantes de hardware tienden cada vez más a mantener en secreto las especificaciones del hardware. Esto dificulta enormemente la tarea de escribir drivers libres para que Linux y Xfree86[6] sean compatibles con el hardware nuevo. Hoy contamos con sistemas libres *y* completos, pero no durarán mucho si no son compatibles con los ordenadores del mañana.

Hay dos formas de enfrentarse a este problema. Los programadores pueden hacer ingeniería inversa para descubrir cómo crear programas compatibles con el hardware. El resto podemos elegir qué hardware será compatible con el software libre. A medida que aumente el número de usuarios de software libre, el secretismo de estas especificaciones se convertirá en una política contraproducente.

Hacer ingeniería inversa es una labor colosal. ¿Contaremos con programadores lo bastante decididos para llevarla a cabo? Sí, siempre que les hayamos convencido de que el software libre es una cuestión de principios y de que los drivers no libres son intolerables. ¿Invertiremos dinero extra, e incluso tiempo extra, para poder utilizar drivers libres? Sí, siempre y cuando se generalice esta voluntad de recuperar nuestra libertad.

Librerías no libres

La librería no libre que opera en un sistema operativo libre constituye una trampa para los desarrolladores de software libre. Las atractivas funciones de la librería son el cebo perfecto; al utilizar la librería, caes en la trampa, porque tu programa no puede integrarse de forma útil en un sistema operativo libre. [Estrictamente hablando, podríamos incluir tu programa, pero éste no podría ejecutarse sin la librería]. Y, lo que es

[6] El Xfree86 es un programa que proporciona un entorno de escritorio que interactúa con tu hardware —ratón, teclado, etc. Funciona en plataformas muy diversas.

peor, en caso de popularizarse un programa que utilice una librería propietaria, podría arrastrar a otros programadores desprevenidos hacia la misma trampa.

El primer ejemplo de este problema se presentó en los años ochenta, con el Motif *toolkit*.[7] Aunque entonces no había sistemas operativos libres, estaba claro qué problema iba a plantearles el Motif más tarde. El Proyecto GNU respondió de dos formas: planteando la necesidad de que los proyectos individuales de software libre fueran compatibles tanto con los toolkit widgets X libres como con Motif, y encargando la creación de un sustituto libre para Motif. La tarea tardó muchos años en concluirse. Sólo en 1997 el LessTif, desarrollado por los Hungry Programmers, fue lo suficientemente potente para la mayoría de las aplicaciones Motif.

Entre 1996 y 1998, otro toolkit no libre de Graphical User Interface (GUI) llamado Qt se incorporó a una notable colección de software libre, el escritorio KDE.

Los sistemas libres GNU/Linux no podían aprovechar el KDE porque no podíamos emplear la librería. A pesar de ello, algunos distribuidores comerciales de sistemas GNU/Linux, bastante flexibles a la hora de mezclar software libre, añadieron el KDE a sus sistemas —lo cual daría lugar a un sistema con más posibilidades y menos libertad. El grupo KDE animó activamente a otros programadores a que utilizasen Qt, mientras que millones de nuevos «usuarios de Linux» ni siquiera sospechaban que pudiera existir un problema al respecto. La situación era desoladora.

La comunidad del software libre reaccionó de dos maneras: GNOME y Harmony.

GNOME, el GNU Network Object Model Environment, es el proyecto de escritorio de GNU. Miguel de Icaza tomó la iniciativa en 1997, y se desarrolló con el apoyo de Red Hat Software. GNOME pretendía proveer prestaciones similares, pero usando exclusivamente software libre. Entraña algunas ventajas técnicas, como la de ser compatible con varios lenguajes, y no sólo el C++. Pero su principal propósito era la libertad, funcionar sin software no libre.

Harmony es una librería sustitutiva compatible, diseñada con el fin de ejecutar software KDE sin recurrir a Qt.

En noviembre de 1998, los desarrolladores de Qt anunciaron un cambio de licencia que, en caso de aplicarse, lo convertiría en software libre. Aunque no podemos estar seguros de esto, creo que el cambio se debió en parte a la firme respuesta de la comunidad ante el problema que planteaba la condición no libre de Qt. [Esta nueva licencia es incómoda y no equitativa, por lo que sigue siendo aconsejable evitar el uso de Qt[8]]

¿Cómo responderemos a la tentación que plantee la próxima librería no libre? ¿Comprenderá la comunidad la necesidad de mantenernos alejados de cualquier trampa que se nos presente en el camino? ¿O renunciaremos a la libertad a cambio de la comodidad, y dar lugar así a un problema mucho mayor? Nuestro futuro depende de nuestra filosofía.

---

[7]Motif es una interfaz gráfica y administrador de ventanas que opera en X Window, un potente sistema gráfico basado en una arquitectura cliente/servidor.

[8]En septiembre de 2000 volvería a publicarse el Qt con GNU GPL, lo que básicamente resolvía el problema.

Patentes de software

La amenaza más seria a la que nos enfrentamos procede de las patentes de software, que pueden introducir algoritmos y funciones fuera del alcance del software libre al menos durante veinte años. Las patentes del algoritmo de compresión LZW se aplicaron en 1983, y todavía no podemos publicar software libre que produzca GIFs adecuadamente comprimidos. En 1998, se suspendió la distribución de un programa libre para producir archivos de audio MP3 comprimidos bajo amenaza de una demanda judicial por patente.

Existen formas de abordar la cuestión de las patentes: buscar pruebas que demuestren la invalidez de una patente o buscar modos alternativos para realizar una tarea. Pero estos métodos funcionan sólo de vez en cuando; cuando fallan ambos, la patente puede resultar en un software libre desprovisto de alguna función necesaria para los usuarios. ¿Qué haremos entonces?

Quienes valoramos el software libre por la libertad que éste entraña seguiremos en la misma línea. Lograremos sacar adelante el trabajo sin funciones patentadas. Pero quienes valoran el software libre porque esperan que sea técnicamente superior se inclinarán por calificarlo de fracaso cuando este software se vea restringido por una patente. De modo que, a pesar de que resulta muy útil discutir la efectividad práctica del modelo de desarrollo de tipo «catedral» y la fiabilidad y potencia de ciertos programas de software libre, debemos ir más allá. Debemos hablar de libertad y de principios.

Documentación libre

La mayor deficiencia de nuestros sistemas operativos no reside en el software, sino en la ausencia de buenos manuales libres para nuestros sistemas. La documentación es una parte esencial de cualquier paquete de software; un paquete importante de software libre sin un buen manual libre que lo acompañe constituye un lastre considerable. Tenemos muchos ejemplos de ello en la actualidad.

La documentación libre, al igual que el software, es una cuestión de libertad, no de precio. Los criterios para el manual libre son bastante parecidos a los del software libre: hay que darles a los usuarios ciertas libertades. Debe autorizarse la redistribución —incluida la venta comercial— en papel y *on line*, de modo que el manual pueda acompañar a todas las copias del programa.

Autorizar su modificación resulta igualmente crucial. Por regla general, no creo que la gente deba tener el derecho de modificar toda clase de artículos y libros. Por ejemplo, no creo que ni tú ni yo estemos obligados a autorizar la modificación de artículos como este, que describe nuestros actos y opiniones.

Pero existe una razón específica de que la libertad para modificar sea un elemento crucial para la documentación relativa al software libre. Cuando los individuos ejercen su derecho a modificar el software, y añadir o cambiar sus funciones, si son lo bastante concienzudos cambiarán asimismo el manual —y así proporcionarán una documentación concisa y útil junto con el programa modificado. Un manual que no permita a los

programadores trabajar concienzudamente y terminar su labor no satisfará las necesidades de la comunidad.

Algunos límites a la incorporación de estas modificaciones no plantean problema alguno, como es el caso de los requisitos establecidos para preservar la advertencia sobre *copyright* del autor original, los términos de distribución o la lista de autores. O aquellos que exigen que las versiones modificadas incluyan la fecha de la modificación, o que incluso prohíben la supresión o alteración de secciones enteras, siempre que éstas no traten sobre temas técnicos. Este tipo de restricciones no plantean un problema porque no impiden al programador concienzudo adaptar el manual para que se ajuste al programa modificado. Dicho de otro modo, no impiden que la comunidad de software libre disfrute plenamente del uso del manual.

Sin embargo, debemos ser capaces de modificar el contenido «técnico» del manual y luego distribuir el resultado en los medios y canales habituales; de lo contrario, las restricciones obstruirán a la comunidad, el manual dejará de ser libre y necesitaremos elaborar uno nuevo.

¿Contarán los desarrolladores de software libre con la conciencia y la determinación para producir una amplia gama de manuales libres? Una vez más, nuestro futuro depende de nuestra filosofía.

Es necesario hablar de libertad

Actualmente, se calcula que existen diez millones de usuarios de sistemas GNU/Linux como Debian GNU/Linux y Red Hat Linux. El software libre ha desarrollado tales ventajas prácticas que está ganando adeptos por razones puramente prácticas.

Las consecuencias positivas de esto son evidentes: un mayor interés por desarrollar software libre, más clientes para las empresas de software libre y una mayor capacidad para alentar a las empresas a desarrollar software libre comercial en lugar de productos de software propietario.

Pero el interés en el software crece a un ritmo superior que la conciencia de la filosofía en que se fundamenta, y esto plantea ciertas dificultades. Nuestra capacidad para hacer frente a los desafíos y las amenazas anteriormente descritos dependerá de nuestra voluntad de mantenernos firmes en nombre de la libertad. Para convencer de ello a nuestra comunidad, habremos de difundir la idea entre los nuevos usuarios que pasen a formar parte de ella.

Pero estamos fracasando: nuestros esfuerzos por atraer a nuevos usuarios a nuestra comunidad superan con creces a nuestras iniciativas a la hora de enseñarles los principios de nuestra comunidad. Debemos dedicarnos a ambos objetivos y compensar nuestros esfuerzos en ambas direcciones.

«Open Source» (código fuente abierto)

La tarea de enseñar a los nuevos usuarios el valor de la libertad se complicó especialmente en 1998, cuando parte de la comunidad decidió abandonar el término «software libre» y empezó a hablar de «software de código abierto».

Los partidarios de este término trataban de evitar la confusión entre «libre» y «gratuito»—un objetivo muy legítimo. Pero otros intentaban dejar a un lado los principios que habían impulsado la creación del software libre y el proyecto GNU, procurando así atraer a los ejecutivos y a los usuarios de empresas, quienes comparten mayoritariamente una ideología que antepone las ganancias económicas a la libertad, a la comunidad, a los principios. De modo que la retórica del «código abierto» se concentra en la posibilidad de crear un software de alta calidad y capacidad, pero rehuye las nociones de libertad, comunidad y principios.

Un claro ejemplo de ello son las revistas «Linux» —están repletas de anuncios de software propietario que funciona con GNU/Linux. Cuando aparezca el próximo Motif, o Qt, ¿advertirán estas revistas a los programadores de que se alejen de ellos, o los anunciarán sin más?

El apoyo de la comunidad empresarial puede contribuir al bien de la comunidad de distintas maneras, siempre que partamos de unas condiciones de igualdad. Pero si nos ganamos su apoyo callándonos lo que pensamos sobre la libertad y los principios, el resultado puede ser desastroso, y sólo se agudizaría el desequilibrio ya existente entre la difusión y la educación cívica.

Los términos «software libre» y «código abierto» describen más o menos la misma categoría de software, pero implican cosas muy distintas acerca del software y sus valores. El Proyecto GNU sigue empleando el término «software libre» para expresar la idea de que la libertad, y no sólo la tecnología, es importante.

¡Inténtalo!

La filosofía de Yoda —«No podemos sólo intentarlo»— suena bien, pero no me sirve. He realizado mi trabajo siempre ansioso ante la perspectiva de que no tuviera suficiente capacidad para ello, sin saber si mi labor bastaría para alcanzar el objetivo deseado. Pero lo intenté de todas formas, porque entre el enemigo y mi ciudad sólo estaba yo. Para mi sorpresa, a veces del éxito obtenido.

En otras ocasiones fracasé. Algunas de mis ciudades han caído. Más tarde descubrí otra ciudad amenazada y me preparé para otra batalla. Con el tiempo, he aprendido a detectar las amenazas y a interponerme entre ellas y mi ciudad, haciendo un llamamiento a otros hackers para unirse a mí.

Hoy en día, a menudo me encuentro que no estoy solo. La visión de un regimiento de hackers manos a la obra constituye una fuente de alivio y de alegría, y pienso que la ciudad sobrevivirá por el momento. Pero con el transcurso de los años los peligros son cada vez mayores, y ahora Microsoft nos tiene en su punto de mira. No podemos

pensar que el futuro de la libertad está asegurado. ¡No os engañéis! Si quieres conservar tu libertad, tienes que estar preparado para defenderla.

Capítulo 2

El Manifiesto GNU[1]

El Manifiesto GNU fue escrito por Richard Stallman en los comienzos del Proyecto GNU, con el fin de pedir participación y apoyo. En los primeros años, se hicieron actualizaciones menores para tomar en cuenta nuevos desarrollos, pero ahora parece ser mejor dejarlo como esta tal y como la mayoría de la gente lo ha visto. Desde entonces, hemos aprendido que es posible ayudar a evitar ciertas confusiones corrientes con un cambio en la selección de palabras, a lo largo de estos se han añadido notas al pie de página para aclarar estas confusiones.

¿Qué es GNU? ¡Gnu No es Unix!

GNU, que significa Gnu No es Unix, es el nombre de un sistema de software completamente compatible con Unix que estoy escribiendo para poder regalarlo[2] libremente a quien pueda utilizarlo. Hay varios voluntarios ayudándome. Son muy necesarias las contribuciones en tiempo, dinero, programas y equipamiento.

Hasta el momento tenemos un editor de texto Emacs con Lisp para escribir comandos de edición, un depurador de código fuente, un generador compatible con yacc, un comunicador y alrededor de 35 utilidades. Un shell (intérprete de comandos) se encuentra casi terminado. Un nuevo compilador portable de C ha sido compilado y será lanzado este año. Existe un kernel inicial pero requiere muchas más características para emular a Unix. Cuando el núcleo y el compilador estén completos, será posible distribuir un sistema GNU apropiado para el desarrollo de programas. Usaremos el procesador de documentos TeX, pero también se está trabajando en un nroff. Usaremos también el sistema libre de ventanas X que se puede migra. Después de esto agregaremos un Lisp portable, un juego Empire, una hoja de cálculo y cientos de otras cosas,

[1] Publicado por primera vez en 1984

[2] Esta expresión era desafortunada. La intención era decir que nadie tendría que pagar por el *permiso* de usar el sistema GNU. Pero las palabras no aclaran, y la gente interpreta a menudo que se dice que las copias de GNU deberán ser siempre distribuidas a un costo bajo o sin costo. Ésta nunca fue la intención; más adelante, el manifiesto menciona la posibilidad de que las compañías provean servicios de distribución con una ganancia. Con posterioridad he aprendido a distinguir cuidadosamente entre «free» [libre] en el sentido de libertad y «free» [gratis] en el sentido del precio. Software libre es software que los usuarios tienen la libertad de distribuir y cambiar. Algunos usuarios pueden obtener copias sin pagar, mientras que otros pagan para obtener copias; y si los fondos ayudan a apoyar la mejora del software, tanto mejor. Lo importante es que todos los que tengan una copia tengan la libertad de cooperar con otros al usarlo.

además de documentación en línea. Esperamos proporcionar, con el tiempo, todas las utilidades que vienen normalmente con un sistema Unix, y más.

GNU será capaz de ejecutar programas de Unix, pero no será idéntico a Unix. Haremos todas las mejoras que sean convenientes y estén basadas en nuestra experiencia con otros sistemas operativos. Concretamente, planeamos tener nombres de archivos más largos, números de versión de archivos, un sistema de archivos a prueba de caídas, tal vez incorporemos un sistema para completar nombres de archivos, soporte en pantalla independiente del tipo de terminal y quizá en un futuro un sistema de ventanas basado en Lisp a través del cual varios programas Lisp y programas ordinarios Unix puedan compartir una sola pantalla. Tanto C como Lisp estarán disponibles como lenguajes de programación. Intentaremos soportar UUCP, Chaosnet del MIT y otros protocolos para comunicación en Internet.

GNU está siendo escrito inicialmente para máquinas de la clase 68000/16000 con memoria virtual, porque éstas son las máquinas en las que es más sencilla su ejecución. El esfuerzo adicional para hacerlo funcionar en máquinas más pequeñas se dejará a alguien que quiera utilizarlo en ellas.

Para evitar una horrible confusión, por favor pronuncie la «G» en la palabra «GNU» cuando se refiera al nombre de este proyecto.[3]

Por qué debo escribir GNU

Considero que la regla de oro requiere que si a mi me gusta un programa, lo debo compartir con otra gente que le guste. Los vendedores de software quieren dividir a los usuarios y seducirlos, haciendo que cada usuario acuerde no compartir su software con otros. Yo rehúso a romper mi solidaridad con otros usuarios de esta manera. No puedo en buena conciencia firmar un acuerdo de no divulgación o un acuerdo de licencia de software. Durante años trabajé dentro del Laboratorio de Inteligencia Artificial para impugnar estas tendencias y otras descortesías, pero al final ellos fueron demasiado lejos: no podía permanecer en una institución donde hiciera estas cosas en contra de mi voluntad.

De manera que para poder continuar usando ordenadores sin vergüenza, he decidido agrupar un cuerpo suficiente de software libre de tal manera que pueda seguir andando sin ninguna clase de software que no sea libre. He dejado el laboratorio de IA para negar al MIT cualquier excusa legal que me prohíba distribuir software GNU.

Por qué GNU será compatible con Unix

Unix no es mi ideal de sistema, pero no es del todo malo. Las características esenciales de Unix parecen ser buenas y pienso que puedo añadir lo que le falta a Unix sin echarlas a perder. Un sistema compatible con Unix sería conveniente para que otras personas puedan adoptarlo.

[3] GNU se pronuncia en inglés de forma muy similar a [4], que significa «nuevo». [*N. del E.*]

Cómo estará disponible GNU

GNU no es de dominio público. Todos tendrán permiso para modificar y redistribuir GNU, pero a ningún distribuidor se le permitirá restringir su redistribución posterior. Es decir, no estarán permitidas modificaciones propietarias. Quiero asegurarme de que todas las versiones de GNU permanezcan libres.

Por qué quieren cooperar muchos otros programadores

He encontrado muchos otros programadores que están entusiasmados con GNU y quieren ayudar.

Muchos programadores están descontentos con la comercialización del software de sistema. Puede permitirles ganar más dinero, pero por lo general les hace sentirse en conflicto con otros programadores, en lugar de sentirse como compañeros. El acto fundamental de amistad entre programadores es el hecho de compartir programas; los acuerdos de marketing usados de forma generalizada esencialmente prohíben a los programadores tratar a sus semejantes como amigos. El comprador de software debe escoger entre la amistad y la obediencia a la ley. Naturalmente, muchos deciden que la amistad es más importante. Pero aquellos que creen en la ley a menudo no se sienten bien con ninguna de las dos opciones. Se vuelven cínicos y piensan que la programación es sólo otra forma de hacer dinero.

Al desarrollar y utilizar GNU en lugar de programas propietarios, nosotros podemos ser hospitalarios con todos y obedecer la ley. Además, GNU sirve como ejemplo de inspiración y como bandera para conminar a otros a unirse a nosotros en el acto de compartir. Esto puede darnos una sensación de armonía que es imposible cuando utilizamos software que no es libre. Porque para cerca de la mitad de los programadores con quienes he hablado, este es un motivo de felicidad importante, que el dinero no puede reemplazar.

Cómo puedes contribuir

Estoy pidiendo a los fabricantes de ordenadores que donen equipos y dinero. Estoy pidiendo donativos en forma de programas y trabajo.

Una consecuencia que esperable si donas equipos es que GNU se ejecutará en ellos desde una etapa temprana. Las máquinas deberán estar completas, con los sistemas listos para su uso, probadas para uso en áreas residenciales y no requerir ventilación o fuentes de energía sofisticadas.

He encontrado que muchos programadores están ansiosos de contribuir trabajando a tiempo parcial para GNU. Para la mayoría de los proyectos esta distribución de trabajos a tiempo parcial sería muy difícil de coordinar. Pero para la tarea particular de reemplazar Unix no existe este problema. Un sistema completo en Unix contiene cientos de utilidades, cada una de las cuales se documenta por separado. Casi todas las especificaciones de la interfaz han sido fijadas para ser compatibles con Unix. Si

cada colaborador puede escribir un sustituto compatible para una sola utilidad Unix, y hacer que funciones apropiadamente en lugar del programa original en un sistema Unix, entonces estas utilidades funcionarán correctamente cuando sean reunidas. Incluso permitiendo que Murphy[5] origine algunos problemas inesperados, el ensamblaje de estos componentes será una tarea factible. (El kernel va a requerir una comunicación más estrecha y un grupo más pequeño y unido trabajará en él).

Si obtengo donativos de dinero, estaré en condiciones de contratar unas cuantas personas a tiempo completo o a tiempo parcial. El sueldo no será alto en relación al estándar de los programadores, pero estoy buscando a gente para quien la construcción de espíritu comunitario tenga tanta importancia como hacer dinero. Considero esto como una manera de dar oportunidad a gente interesada para que dediquen todas sus energías a trabajar en GNU, evitando la necesidad de ganarse la vida de otra manera.

Por qué se beneficiarán todos los usuarios de ordenadores

Una vez que se haya escrito GNU, todos podremos obtener un buen sistema de software libre, al igual que obtenemos aire.[6]

Esto significa mucho más que el simple ahorro del precio de una licencia Unix. Significa que se evitará mucho del derroche de esfuerzos en la duplicación de la programación de sistemas. Este esfuerzo puede enfocarse mejor para hacer avanzar el estado de la técnica.

Los códigos completos del sistema estarán disponibles para todo el mundo. Como resultado, un usuario que necesite modificar el sistema será siempre libre de hacerlo por sí mismo, o de contratar a cualquier programador o empresa disponible para hacerlo por él. Los usuarios no estarán ya a merced de un programador o una empresa que sea dueña del código fuente y que sea la única en posición de realizar modificaciones.

Las escuelas podrán proporcionar un ambiente bastante más educativo, animando todos los estudiantes a estudiar y mejorar el código del sistema. En el laboratorio de programación de Harvard solían tener la política de que ningún programa podía ser instalado en el sistema si su código fuente no estaba a disposición del público; la ejercitaban a base de negarse a instalar ciertos programas. En buena medida, yo me inspiré en esto.

Finalmente, el lastre de considerar quién es dueño de qué sistema de software y de lo que está o no está permitido hacer con él, habrá desaparecido.

Los acuerdos para obligar a la gente a pagar por utilizar un programa, incluyendo la licencia de copias, siempre incurren en un tremendo coste para la sociedad por los aparatosos mecanismos necesarios para determinar cuánto —esto es, qué programas—

[5]Esta es una referencia a «La Ley de Murphy» una ley humorística que dice que si es posible que algo pueda ir mal, irá mal.

[6]Este es otro sitio en donde me equivoqué al no distinguir cuidadosamente entre los dos diferentes significados de «libre». La afirmación tal como está escrita no es falsa: uno puede obtener copias gratuitas de software GNU, ya sea de sus amistades o a través de la Internet. Pero se está sugiriendo una idea errónea.

debe pagar una persona. Sólo un estado policial puede forzar a obedecer a todo el mundo. Considérese una estación espacial en donde el aire debe fabricarse con un gran coste: cobrar a cada respirador por litro de aire quizá sea justo, pero el uso continuo de una máscara de aire con medidor todo el día y toda la noche sería intolerable, aunque todos pudieran permitirse pagar su consumo de aire. Tener cámaras de televisión por todos lados para ver si se quita uno la máscara sería el colmo. Es mejor costear la planta de aire con un impuesto por persona y desechar las máscaras.

Copiar por completo o una parte de un programa es tan natural para un programador como respirar y le es igual de productivo. Debiera ser igualmente libre.

Algunas objeciones fácilmente rebatibles a los objetivos de GNU:

—*«Nadie lo usará debido a que es gratuito, ya que esto significa que no cuenta con ninguna asistencia».*

—*«Se tiene que cobrar por el programa para pagar el servicio de soporte.»*

Si la gente prefiriera pagar por GNU y su servicio en vez de obtener GNU libre y sin servicio, una empresa que ofrezca servicio a las personas que obtuvieron GNU libre debiera tener beneficios.

Debemos distinguir entre asistencia bajo la forma de un verdadero trabajo de programación y lo que es meramente llevar de la mano al usuario. En relación a la primera no se puede depender del vendedor de software. Si tu problema no es compartido por un número suficiente de clientes, el vendedor lo ignorará.

Si tu negocio requiere poder confiar en la asistencia, la única manera es tener el código fuente y todas las herramientas necesarias. De este modo, puedes contratar a cualquier persona disponible para corregir el problema; y no estar a merced de ningún individuo. Con Unix, el precio del código fuente deja esta posibilidad fuera de cualquier consideración para la mayoría de los negocios. Con GNU esto será sencillo. Es posible que todavía no haya ninguna persona competente disponible, pero este problema no se le puede imputar a los acuerdos de distribución. GNU no elimina todos los problemas, sólo algunos de ellos.

Mientras tanto, los usuarios que no saben de ordenadores necesitan que se les lleve de la mano: hacer cosas por ellos que ellos mismos podrían hacer fácilmente, pero que no saben cómo hacerlas.

Estos servicios podrán ser proporcionados por compañías que vendan solamente servicios de asesoría y de reparación. Si es verdad que los usuarios prefieren gastar dinero y obtener un producto con servicio, estarán igualmente de acuerdo en adquirir el servicio habiendo obtenido el producto de forma gratuita. Las empresas de servicios competirán en calidad y precio; los usuarios no estarán atados a ninguna en particular. Entre tanto, aquellos de nosotros que no necesitemos servicios debemos poder usar el programa sin pagar por el servicio.

—*«No se puede llegar a mucha gente sin publicidad y uno debe cobrar por el programa para mantener esto».*

—*«No tiene sentido dar publicidad a un programa que la gente puede obtener gratuitamente».*

Existen varias formas de publicidad gratuita o de bajo costo que pueden usarse para informar a numerosos usuarios de ordenadores acerca de algo como GNU. Pero quizá sea verdad que se puede llegar a más usuarios de microordenadores a través de la publicidad. Si realmente es así, un negocio que haga publicidad del servicio de copia y envío de GNU por un precio, debería ser lo suficientemente exitoso como para pagar como mínimo su publicidad. De esta forma, sólo los usuarios que se beneficien de la publicidad pagarán por ella.

Por otro lado, si mucha gente obtiene GNU a través de sus amistades y estas empresas no resultan ser un buen negocio, esto demostraría que la publicidad no era realmente necesaria para divulgar GNU. ¿Por qué será que los defensores del libre mercado no quieren permitir que el libre mercado decida esto?[7]

—*«Mi compañía necesita un sistema operativo propietario para tener una ventaja competitiva».*

GNU sacará al software de sistema operativo del entorno de la competencia. Usted no podrá obtener una ventaja en esta área, pero tampoco la competencia podrá tenerla frente a usted. Usted y ellos competirán en otras áreas, mientras se benefician mutuamente en ésta. Si tu negocio es vender un sistema operativo, no te gustará GNU, pero ese es tu problema. Si tu negocio es otra cosa, GNU puede salvarlo de ser arrojado al costoso negocio de la venta de sistemas operativos.

Me gustaría ver que el desarrollo de GNU se mantuviera gracias a donaciones de algunos fabricantes y usuarios, reduciendo el costo para sí mismos.[8]

—*«¿No merecen los programadores una recompensa por su creatividad?»*

Si hay algo que merezca una recompensa es la contribución social. La creatividad puede considerarse una contribución social, pero sólo si la sociedad es libre de aprovechar sus resultados. Si los programadores merecen ser recompensados por la creación de programas innovadores, bajo esta misma lógica deben ser castigados si restringen el uso de estos programas.

—*«¿No debería tener el programador la opción de pedir una recompensa por su creatividad?»*

No hay nada que objetar en querer un pago por el trabajo, o por buscar maximizar los propios ingresos, siempre y cuando no se utilicen medios que sean destructivos. Pero las formas a las que estamos acostumbrados actualmente en el campo del software se basan en la destrucción.

Extraer dinero de los usuarios de un programa mediante la restricción de su uso resulta destructivo porque las restricciones reducen la cantidad y las formas en que el

[7] La Free Software Foundation obtiene la mayor parte de sus fondos de su servicio de distribución, aunque se trata de una organización sin ánimo de lucro y no de una compañía. Si nadie opta por obtener copias haciendo su pedido a la FSL, ésta no estará en disposición de hacer su trabajo. Pero esto no significa que las restricciones propietarias sean justificables para obligar a todos a pagar. Si una pequeña fracción de todos los usuarios pide sus copias a la FSL, será suficiente para mantener a la FSL a flote. Por lo tanto, tenemos que pedir a los usuarios que opten por apoyarnos de esta forma. ¿Has hecho ya tu parte?

[8] Recientemente, un grupo de compañías de informática ha reunido fondos para apoyar el mantenimiento del compilador C GNU.

programa puede ser utilizado. Esto reduce la cantidad de riqueza que la humanidad obtiene del programa. Cuando se opta deliberadamente por la restricción, las consecuencias dañinas son la destrucción deliberada.

La razón por la que un buen ciudadano no utiliza estos medios destructivos para volverse más rico es que si todos lo hicieran, podríamos empobrecer por medio de una mutua destrucción. Esto es ética kantiana, o la Regla de Oro. Como no me gustan las consecuencias que resultarían si todos acapararan información, debo considerar como erróneo que alguien lo haga. Específicamente, el deseo de ser recompensado por la creatividad de uno no justifica privar al mundo en general de toda o parte de esa creatividad.

—*«¿No pasarán hambre los programadores?»*

Podría responder que no se fuerza a nadie a ser programador. Casi ninguno de nosotros puede lograr obtener dinero por estar de pie en la calle y hacer muecas. Pero no estamos, como resultado, condenados a estar toda nuestra vida de pie en la calle haciendo muecas y padeciendo hambre. Nos dedicamos a otra cosa.

Sin embargo, ésta es una respuesta errónea porque acepta la suposición implícita del interrogador: que sin la propiedad del software a los programadores no se les puede pagar un céntimo. En este supuesto es todo o nada.

La verdadera razón por la que los programadores no se morirán de hambre es porque aún es posible que se les pague por programar; sólo que no se les pagará tanto como en la actualidad.

Restringir la copia no es la única forma para hacer negocios con el software. Es la forma más común porque es de la que se obtiene más dinero. Si ésta se prohibiera o fuese rechazada por el comprador, el negocio del software se movería hacia otras formas de organización que actualmente no se usan tan a menudo. Hay siempre muchos modos para organizar cualquier tipo de negocio.

Probablemente la programación no será tan lucrativa bajo esta nueva forma como lo es actualmente. Pero esto no es un argumento en contra del cambio. No se considera una injusticia que los dependientes de tiendas obtengan los salarios que ganan actualmente. Si los programadores ganaran lo mismo, no será tampoco una injusticia. (En la práctica ellos ganarán considerablemente más).

—*«¿No tiene derecho la gente a controlar cómo se usa su creatividad?»*

El «control sobre el uso de las ideas de uno» realmente constituye control sobre las vidas de otras personas; y normalmente se usa para dificultar más sus vidas.

La gente que ha estudiado cuidadosamente el tema de los derechos de propiedad intelectual —como los abogados— dice que no hay un derecho intrínseco a la propiedad intelectual. Los tipos supuestos de derechos de propiedad intelectual que reconoce el gobierno fueron creados por actos específicos de legislación con propósitos específicos.

Por ejemplo, el sistema de patentes fue establecido para animar a los inventores a revelar los detalles de sus inventos. Su propósito fue ayudar a la sociedad y no tanto ayudar a los inventores. El periodo de vida de 17 años para una patente era corto

comparado con la tasa de desarrollo de la técnica. Dado que las patentes sólo son relevantes para los fabricantes, para quienes el costo y esfuerzo de un acuerdo de licencia son pequeños comparados con la puesta en marcha de la producción, las patentes a menudo no hacen mucho daño. No obstruyen a la mayoría de los individuos que usan productos patentados.

La idea del copyright no existía en tiempos antiguos, cuando los autores frecuentemente copiaban bastantes obras de otros autores en obras de no ficción. Esta práctica era útil, y ha sido la única forma de que las obras de muchos autores, aunque sólo sea en parte, hayan sobrevivido. El sistema de derechos de autor fue creado expresamente con el propósito de promover la autoría. En el ámbito para el que se inventó —libros, que sólo podían ser copiados económicamente en una imprenta— hacía muy poco daño y no obstruía a la mayor parte de los individuos que leían los libros.

Todos los derechos de propiedad intelectual son únicamente licencias que otorga la sociedad porque se pensaba, correcta o equivocadamente, que la sociedad en conjunto se beneficiaría al ser otorgados. Pero en cualquier situación particular, necesitamos preguntarnos: ¿nos beneficia haber otorgado tal licencia? ¿Qué tipo de acto estamos permitiendo que haga una persona?

El caso de los actuales programas es muy diferente al de los libros de hace cien años. El hecho de que la forma más sencilla de copiar un programa sea de un vecino a otro, el hecho de que un programa sea tanto el código fuente como el código objeto, siempre distintos, y el hecho de que el programa sea usado y no leído y disfrutado, se combinan para crear una situación en la que una persona que hace valer un copyright está dañando a la sociedad en su conjunto tanto materialmente como espiritualmente; nadie debería hacerlo a pesar de que la ley se lo permita.

—*«La competencia hace que las cosas se hagan mejor».*

El paradigma de la competencia es una carrera: al premiar al ganador, estamos alentando a todos a correr más rápido. Cuando el capitalismo realmente trabaja de esta manera, hace un buen trabajo; pero sus partidarios están equivocados al asumir que siempre funciona así. Si los corredores olvidan por qué se otorga el premio y se centran en ganar sin importar cómo, pueden encontrar otras estrategias —como atacar a los otros corredores. Si los corredores se enredan en una pelea a puñetazos, todos llegarán tarde a la meta.

El software propietario y secreto es el equivalente moral de los corredores en una pelea a puñetazos. Es triste decirlo, pero el único árbitro que tenemos no parece estar en contra de las peleas; sólo las regula —«por cada 10 yardas que corras, tienes derecho a un disparo». Lo que debería hacer es separarlos y penalizar a los corredores por el solo hecho de intentar pelear.

—*«¿No dejarán todos de programar si no hay un incentivo monetario?»*

De hecho, mucha gente va a programar sin absolutamente ningún incentivo monetario. La programación tiene una fascinación irresistible para algunas personas, generalmente para las mejores en el ramo. No hay escasez de músicos profesionales que sigan en lo suyo aunque no tengan esperanzas de ganarse la vida de esta forma.

Sin embargo en realidad esta pregunta, aun cuando se hace muchas veces, no es apropiada a la situación. El pago a los programadores no va a desaparecer, sólo se va a reducir. La pregunta correcta es, ¿alguien programará con la reducción en el incentivo monetario? Mi experiencia muestra que sí lo harán.

Durante más de diez años, varios de los mejores programadores del mundo trabajaron en el Laboratorio de Inteligencia Artificial por mucho menos dinero del que podían ganar en otras partes. Recibieron varios tipos de recompensas no económicas: fama y aprecio, por ejemplo. Y la creatividad también se disfruta, es un premio en sí misma.

Luego la mayoría se fue cuando se les ofreció la oportunidad de hacer ese mismo trabajo interesante por mucho dinero.

Lo que muestran los hechos es que la gente programa por razones distintas a la de la riqueza; pero si se les da una oportunidad de hacer también mucho dinero, ésta entrará en sus expectativas y la van a exigir. Las organizaciones que pagan poco no podrán competir con las que pagan mucho, pero no tendría que irles tan mal si las que pagan mucho fueran prohibidas.

—*«Necesitamos a los programadores desesperadamente. Si ellos nos pidieran que dejemos de ayudar a nuestro prójimo, tendríamos que obedecer».*

Uno nunca está tan desesperado como para tener que obedecer este tipo de exigencia. Recuerda: millones para nuestra defensa, ¡pero ni un céntimo para tributos!

—*«Los programadores necesitan tener alguna forma de ganarse la vida».*

A corto plazo, esto es verdad. Sin embargo, hay bastantes maneras de que los programadores puedan ganarse la vida sin vender el derecho a usar un programa. Esta manera es frecuente ahora porque es la que les da a los programadores y hombres de negocios más dinero, no porque sea la única forma de ganarse la vida. Es fácil encontrar otras formas, si quieres encontrarlas.

He aquí unos cuantos ejemplos:

- Un fabricante introduce un nuevo ordenador y pagará por adecuar los sistemas operativos al nuevo hardware.
- La venta de enseñanza, los servicios de asistencia y mantenimiento también pueden dar trabajo a programadores.
- La gente con ideas nuevas podrá distribuir programas como freeware, pidiendo donativos a usuarios satisfechos, o vendiendo servicios de asistencia. Yo he conocido a personas que ya trabajan así con éxito.
- Los usuarios con necesidades comunes pueden formar un grupo de usuarios y pagar sumas de dinero. Un grupo contratará a empresas de programación para escribir programas que a los miembros del grupo les gustaría utilizar.

Todo tipo de desarrollo puede financiarse con un «impuesto de software»:

- Supón que todos los que compren un ordenador tengan que pagar un tanto por ciento de su precio como impuesto de software. El gobierno entrega este dinero a una agencia como la Fundación Nacional de la Ciencia para que lo emplee en el desarrollo de software.
- Pero si el comprador del ordenador hace por sí mismo un donativo para el desarrollo de software puede verse exento de este impuesto. Puede donar al proyecto de su elección—a menudo, elegido porque espera utilizar los resultados tan pronto como se haya completado. Puede tomar un crédito por cada cantidad de donativo hasta la totalidad del impuesto que tenía que pagar.
- La tasa total de impuesto podrá ser decidida por un voto de los contribuyentes al impuesto, sopesada de acuerdo con la cantidad sobre la que se aplicará el impuesto.

Las consecuencias:

- La comunidad de usuarios de ordenadores apoya el desarrollo del software.
- Esta comunidad decide qué nivel de apoyo se necesita.
- Los usuarios a quienes les importa a qué proyectos se destine su parte pueden escogerlos por sí mismos.

A largo plazo, hacer programas libres es un paso hacia el mundo post-escasez, donde nadie tendrá que trabajar duro para ganarse la vida. La gente será libre para dedicarse a actividades entretenidas, como la programación, después de haber dedicado diez horas obligatorias a la semana a las tareas requeridas, como legislar, el asesoramiento familiar, la reparación de robots y la exploración de asteroides. No habrá necesidad de ganarse la vida mediante la programación.

Hemos alcanzado ya una gran reducción de la cantidad de trabajo que la sociedad en su conjunto debe realizar para mantener su productividad actual, pero sólo un poco de ésta reducción se ha traducido en descanso para los trabajadores, dado que hay mucha actividad no productiva que se requiere para acompañar a la actividad productiva. Las causas principales de esto son la burocracia y las luchas isométricas contra la competencia. El software libre reducirá en gran medida estos drenajes en el área de producción de software. Debemos hacer esto para lograr así avances técnicos en la productividad que se traduzcan en menos trabajo para nosotros.

Capítulo 3
La definición de software libre[1]

Conservamos esta definición de software libre para expresar claramente el verdadero significado de los programas de software libre.

El «software libre» es una cuestión de libertad, no de precio. Para comprender este concepto, debemos pensar en la acepción de libre como en «libertad de expresión» y no como en «barra libre de cerveza».

Con software libre nos referimos a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. Nos referimos especialmente a cuatro clases de libertad para los usuarios de software:

- Libertad 0: la libertad para ejecutar el programa sea cual sea nuestro propósito.
- Libertad 1: la libertad para estudiar el funcionamiento del programa y adaptarlo a tus necesidades —el acceso al código fuente es condición indispensable para esto.
- Libertad 2: la libertad para redistribuir copias y ayudar así a tu vecino.
- Libertad 3: la libertad para mejorar el programa y luego publicarlo para el bien de toda la comunidad —el acceso al código fuente es condición indispensable para esto.

Software libre es cualquier programa cuyos usuarios gocen de estas libertades. De modo que deberías ser libre de redistribuir copias con o sin modificaciones, de forma gratuita o cobrando por su distribución, a cualquiera y en cualquier lugar. Gozar de esta libertad significa, entre otras cosas, no tener que pedir permiso ni pagar para ello.

Asimismo, deberías ser libre para introducir modificaciones y utilizarlas de forma privada, ya sea en tu trabajo o en tu tiempo libre, sin siquiera tener que mencionar su existencia. Si decidieras publicar estos cambios, no deberías estar obligado a notificárselo a ninguna persona ni de ninguna forma en particular.

La libertad para utilizar un programa significa que cualquier individuo u organización podrán ejecutarlo desde cualquier sistema informático, con cualquier fin y sin la obligación de comunicárselo subsiguientemente ni al desarrollador ni a ninguna entidad en concreto.

[1] Escrito originalmente en 1996

La libertad para redistribuir copias supone incluir las formas binarias o ejecutables del programa y el código fuente tanto de las versiones modificadas como de las originales —la distribución de programas en formato ejecutable es necesaria para su adecuada instalación en sistemas operativos libres. No pasa nada si no se puede producir una forma ejecutable o binaria —dado que no todos los lenguajes pueden soportarlo—, pero todos debemos tener la libertad para redistribuir tales formas si se encuentra el modo de hacerlo.

Para que las libertades 2 y 4 —la libertad para hacer cambios y para publicar las versiones mejoradas— adquieran significado, debemos disponer del código fuente del programa. Por consiguiente, la accesibilidad del código fuente es una condición necesaria para el software libre.

Para materializar estas libertades, deberán ser irrevocables siempre que no cometamos ningún error; si el desarrollador del software pudiera revocar la licencia sin motivo, ese software dejaría de ser libre.

Sin embargo, ciertas normas sobre la distribución de software libre nos parecen aceptables siempre que no planteen un conflicto con las libertades centrales. Por ejemplo, el copyleft, *grosso modo*, es la norma que establece que, al redistribuir el programa, no pueden añadirse restricciones que nieguen a los demás sus libertades centrales. Esta norma no viola dichas libertades, sino que las protege.

De modo que puedes pagar o no por obtener copias de software libre, pero independientemente de la manera en que las obtengas, siempre tendrás libertad para copiar, modificar e incluso vender estas copias.

El software libre no significa que sea «no comercial». Cualquier programa libre estará disponible para su uso, desarrollo y distribución comercial. El desarrollo comercial del software libre ha dejado de ser excepcional y de hecho ese software libre comercial es muy importante.

Las normas sobre el empaquetamiento de una versión modificada son perfectamente aceptables siempre que no restrinjan efectivamente tu libertad para publicar versiones modificadas. Por la misma razón, serán igualmente aceptables aquellas normas que establezcan que «si distribuyo el programa de esta forma, deberás distribuirlo de la misma manera» —cabe destacar que esta norma te permite decidir si publicar o no el programa. También admitimos la posibilidad de que una licencia exija enviar una copia modificada y distribuida de un programa a su desarrollador original.

En el proyecto GNU, utilizamos el «copyleft» para proteger legalmente estas libertades. Pero también existe software libre sin copyleft. Creemos que hay razones de peso para recurrir al copyleft, pero si tu programa, software libre, carece de él, todavía tendremos la opción de seguir utilizándolo.

A veces la normativa gubernamental de control de las exportaciones y las sanciones comerciales pueden constreñir tu libertad para distribuir copias a nivel internacional. Los desarrolladores de software no tienen el poder para eliminar o invalidar estas restricciones, pero lo que sí pueden y deben hacer es negarse a imponer estas condiciones de uso al programa. De este modo, las restricciones no afectarán a las actividades y a los individuos fuera de la jurisdicción de estos gobiernos.

Cuando hablamos de software libre, es preferible evitar expresiones como «regalar» o «gratis», porque entonces caeremos en el error de interpretarlo como una mera cuestión de precio y no de libertad. Términos de uso frecuente como el de «piratería» encarnan opiniones que esperamos no compartas. Véase el apartado de «Palabras que conviene evitar» para una discusión sobre estos términos. Tenemos disponible también una lista de traducciones de «software libre» en distintos idiomas.

Por último, señalaremos que los criterios descritos para definir el software libre requieren una profunda reflexión antes de interpretarlos. Para decidir si una licencia de software específica puede calificarse de licencia de software libre, nos basaremos en dichos criterios y así determinaremos si se ajusta al espíritu y a la terminología precisa. Si una licencia incluye restricciones desmedidas, la rechazamos aun cuando nunca predijimos esta cuestión al establecer nuestros criterios. En ocasiones, ciertas condiciones en una licencia pueden plantear un problema que requiera un análisis exhaustivo, lo que significa incluso debatir el tema con un abogado, antes de decidir si dichas condiciones son aceptables. Cuando llegamos a una solución sobre un problema nuevo, a menudo actualizamos nuestros criterios para hacer más fácil la consideración de que licencias están cualificadas y cuáles no.

Si estás interesado en cualificar una licencia específica como licencia de software libre, visita `http://www.gnu.org/licenses/license-list.html`. Si la licencia no aparece en la lista, envíanos un correo electrónico a `licensing@gnu.org`.

Capítulo 4

Por qué el software no debe tener propietarios[1]

Las tecnologías digitales de la información ayudan al mundo haciendo que sea más fácil copiar y modificar información. Los ordenadores prometen hacer esto de forma más sencilla para todos.

No todo el mundo quiere que esto sea más fácil. El sistema de copyright permite que los programas de software tengan «propietarios», la mayor parte de los cuales pretende privar al resto del mundo del beneficio potencial del software. Los propietarios desearían ser los únicos que pueden copiar y modificar el software que usamos.

El sistema de copyright se desarrolló con la imprenta —una tecnología usada para la producción masiva de copias. El copyright se ajustaba bien a esta tecnología puesto que restringía sólo a los productores de copias en masa. No privaba de libertad a los lectores de libros. Un lector cualquiera, que no poseyera una imprenta, sólo podía copiar libros con pluma y tinta, y a pocos lectores se les ponía un pleito por ello.

Las tecnologías digitales son más flexibles que la imprenta: cuando la información adopta forma digital, puedes copiarla fácilmente para compartirla con otros. Es precisamente esta flexibilidad la que se ajusta mal a un sistema como el del copyright. Esa es la razón del incremento de medidas odiosas y draconianas ahora usadas para hacer cumplir el copyright del software. Toma, por ejemplo, estas cuatro prácticas de la Software Publishers Association, SPA [Asociación de Editores de Software]:

- Propaganda masiva afirmando que está mal desobedecer a los propietarios para ayudar a un amigo.
- Solicitar a la gente que se convierta en soplona para delatar a sus colegas y compañeros de trabajo.
- Redadas (con ayuda policial) en oficinas y escuelas, en las que se dice a la gente que debe probar que es inocente de hacer copias ilegales.

[1]Escrito originalmente en 1996

- El proceso judicial —iniciado por el gobierno de los EEUU, a petición de la AES— a personas como David LaMacchia del MIT, [2] no por copiar software —no se le acusa de copiarlo—, sino meramente por dejar sin vigilancia equipos de copia y no controlar su uso.

Cada una de estas cuatro costumbres se asemejan a aquellas usadas en la antigua Unión Soviética, donde todas las máquinas de copiar tenían un vigilante para impedir que se hicieran copias prohibidas, y donde las personas tenían que copiar información en secreto y pasarla de mano a mano como *samizdat*. Por supuesto hay una diferencia: el motivo para el control de información en la Unión Soviética era político; en los EEUU el motivo es el beneficio económico. Pero son las acciones las que nos afectan, no el motivo. Cualquier intento de coartar el hecho de que se comparta la información, sin importar por qué, lleva a los mismos métodos y a la misma dureza.

Los propietarios hacen uso de distintos argumentos para que se les conceda el control de cómo usamos la información:

Insultos

Los propietarios usan palabras difamatorias como «piratería» y «robo», al igual que terminología experta como «propiedad intelectual» y «daño», para sugerir una cierta línea de pensamiento al público —una analogía simplona entre los programas y los objetos físicos.

Nuestras ideas e intuiciones acerca de la propiedad sobre los objetos materiales tratan acerca de si es justo *privarle a alguien de un objeto*. No se aplican directamente a *hacer copias de algo*. Pero los propietarios nos piden que apliquemos estas ideas de todas formas.

Exageración

Los propietarios dicen que sufren un «daño» o «pérdida económica» cuando los usuarios copian programas por su cuenta. Pero copiar no tiene un efecto directo sobre el propietario, y no hace daño a nadie. El propietario sólo puede perder si la persona que hizo la copia hubiese pagado por otra del propietario en su lugar.

Un poco de reflexión muestra que la mayoría de esas personas no habrían comprado copias. Aun así los propietarios calculan sus «pérdidas» como si todos y cada uno hubiesen comprado una copia. Esto es una exageración —por decirlo de forma suave.

La ley

Los propietarios a menudo describen el estado actual de la ley, así como las duras sanciones con las que nos amenazan. Implícita en este enfoque va la sugerencia de que

[2] El 27 de enero de 1995 el caso de David LaMacchia fue desestimado, sin que se haya apelado todavía esta decisión.

la ley actual refleja un punto de vista moral incuestionable —y aun así al mismo tiempo, se nos insta a considerar estas sanciones como hechos naturales por los que no se puede responsabilizar a nadie.

Esta línea de persuasión no está diseñada para reafirmar el pensamiento crítico; está concebida para reforzar un camino mental ya trazado.

Es evidente que las leyes no distinguen lo que está bien de lo que está mal. Todo americano debería saber que, hace cuarenta años, en muchos estados iba contra la ley que una persona de raza negra se sentase en la parte frontal del autobús; pero solamente los racistas dirían que sentarse ahí no estaba bien.

Derecho natural

Los autores a menudo apelan a una conexión especial con los programas que han escrito y añaden que, en consecuencia, sus deseos e intereses respecto al programa simplemente prevalecen sobre los de cualquier otra persona —o incluso sobre los del resto del mundo. (Normalmente son las empresas, no los autores, los que detentan el copyright sobre el software, pero se espera de nosotros que ignoremos esta diferencia.)

Para los que presentan esto como un axioma ético —el autor es más importante que tú— sólo les puedo decir que yo mismo, un notable autor de software, lo considero una tontería.

Sin embargo, la gente, por lo general, sólo suele sentir alguna afinidad hacia las pretensiones basadas en el derecho natural debido a dos razones.

Una razón viene de una analogía forzada entre el software y los objetos materiales. Cuando yo cocino espaguetis, me quejo si otra persona se los come, porque entonces yo ya no me los puedo comer. Su acción me perjudica exactamente tanto como lo que le beneficia a él; sólo uno de nosotros se puede comer los espaguetis, así que la pregunta sería, ¿quién? La más mínima distinción entre nosotros es suficiente para inclinar la balanza ética.

Pero el hecho de que tú ejecutes o modifiques un programa que yo he escrito te afecta a ti directamente y a mí sólo indirectamente. Si tú le das una copia a tu amigo te afecta a ti y a tu amigo mucho más que lo que me afecta a mí. Yo no debería tener el poder de decirte que no hagas estas cosas. Nadie debería.

La segunda razón es que a la gente se le ha dicho que el derecho natural de los autores sen una tradición indiscutida y aceptada en nuestra sociedad.

Desde un punto de vista histórico, lo cierto es lo contrario. La idea del derecho natural de los autores fue propuesta y decididamente rechazada cuando se redactó la Constitución de los EEUU. Ésa es la razón por la que la Constitución sólo *permite* un sistema de copyright y no *obliga a que exista otro*; por esa razón dice que el copyright debe ser temporal. Establece asimismo que el propósito del copyright es promocionar el progreso —no recompensar a los autores. El copyright recompensa a los autores en cierta medida, y a los editores más, pero está concebido como un medio para modificar su comportamiento.

La tradición realmente establecida de nuestra sociedad es que el copyright recorta los derechos naturales del público —y que esto sólo se puede justificar por el bien del público.

Economía

El último argumento que se emplea para justificar la existencia de propietarios de software es que esto conduce a la producción de más software.

A diferencia de los demás, éste argumento por lo menos adopta un enfoque legítimo sobre el tema. Se basa en un objetivo válido —satisfacer a los usuarios de software. Y empíricamente está claro que la gente producirá más de algo si se les paga bien por ello.

Pero el argumento económico tiene un defecto: se basa en la presunción de que la diferencia es sólo cuestión de cuánto dinero debemos pagar. Asume que la «producción de software» es lo que queremos, tenga el software propietarios o no.

La gente acepta gustosamente esta presunción por que está de acuerdo con nuestra experiencia acerca de los objetos materiales. Consideremos por ejemplo un bocadillo. Es posible que puedas conseguir un bocadillo equivalente ya sea gratis, ya por un precio. Si es así, la cantidad que pagas es la única diferencia. Tanto si lo tienes que comprar como si no, el bocadillo tiene el mismo sabor, el mismo valor nutricional, y en cualquier caso te lo puedes comer sólo una vez. El hecho de si el bocadillo lo obtienes de un propietario o no, no puede afectar directamente a nada más que la cantidad de dinero que tienes después.

Esto es cierto para cualquier objeto material —el hecho de que tenga o no tenga propietario no afecta directamente a lo que *es*, o a lo que puedas hacer con ello si lo adquieres.

Pero si un programa tiene un propietario, esto afecta en gran medida a lo que es, y a lo que puedes hacer con una copia si la compras. La diferencia no es sólo una cuestión de dinero. El sistema de propietarios de software incentiva a los propietarios de software a producir algo —pero no lo que la sociedad realmente necesita. Y causa una contaminación ética intangible que nos afecta a todos.

¿Qué es lo que la sociedad necesita? Necesita información que esté verdaderamente a disposición de sus ciudadanos —por ejemplo, programas que la gente pueda leer, arreglar, adaptar, y mejorar, no solamente ejecutar. Pero lo que los propietarios de software ofrecen de forma característica es una caja negra que no podemos ni estudiar ni modificar.

La sociedad también necesita libertad. Cuando un programa tiene un propietario, los usuarios pierden la libertad de controlar una parte de sus propias vidas.

Y sobre todo una sociedad necesita incentivar el espíritu de cooperación entre sus ciudadanos. Cuando los propietarios de software nos dicen que ayudar a nuestro vecino de una manera natural es «piratería», están contaminando el espíritu cívico de nuestra sociedad.

Por eso decimos que el software libre es una cuestión de libertad, no de precio.

El argumento económico para justificar la propiedad es erróneo, pero la cuestión económica es real. Algunas personas escriben software útil por el placer de escribirlo o por admiración y amor; pero si queremos más software del que esas personas escriben, necesitamos conseguir fondos.

Desde hace ya diez años, los desarrolladores de software libre han probado varios métodos para encontrar fondos, con algo de éxito. No hay necesidad de hacer rico a nadie; los ingresos medios de una familia norteamericana, alrededor de 35.000 dólares anuales, demuestran ser incentivo suficiente para muchos trabajos que son menos satisfactorios que programar.

Durante años, hasta que una beca lo hizo innecesario, yo me ganaba la vida realizando mejoras a medida sobre software libre que yo había escrito. Cada mejora se añadía a la versión estándar lanzada y así, eventualmente, se ponían a disposición del público en general. Los clientes me pagaban para que trabajase en las mejoras que ellos querían, en lugar de en las características que yo habría considerado como la máxima prioridad.

La Free Software Foundation (FSF), una entidad sin ánimo de lucro exenta de impuestos para el desarrollo de software libre, consigue fondos mediante la venta de CD-ROMs de GNU, camisetas, manuales y distribuciones «deluxe», (que los usuarios son siempre libres de copiar y modificar), así como mediante donaciones. Ahora cuenta con un equipo de cinco programadores y tres empleados que se encargan de los pedidos por correo.

Algunos desarrolladores de software libre ganan dinero mediante la venta de servicios de soporte. Cygnus Support,[3] que cuenta con alrededor de 50 empleados [en 1994, cuando se escribió este artículo], estima que en torno al 15 por ciento de la actividad de su equipo es desarrollo de software libre —un porcentaje respetable para una compañía de software.

Algunas compañías, incluyendo Intel, Motorola, Texas Instruments y Analog Devices, han unido esfuerzos para financiar el desarrollo continuado del compilador GNU para el lenguaje C. Mientras, el compilador GNU para el lenguaje Ada está siendo financiado por la Fuerza Aérea de los EEUU, que cree que ésta es la manera más efectiva de conseguir un compilador de alta calidad. (La financiación de la Fuerza Aérea se acabó hace algún tiempo; el Compilador GNU de Ada está ahora en servicio, y su mantenimiento se financia comercialmente)

Todos estos ejemplos son pequeños; el movimiento de software libre es aún pequeño y aún joven. Pero el ejemplo de las radios mantenidas-por-la-audiencia en los EE.UU muestra que es posible mantener una actividad grande sin forzar a cada usuario a pagar.

Como un usuario de informática de hoy en día, te puedes encontrar usando un programa propietario. Si un amigo te pide hacer una copia, estaría mal negarse a ello. La cooperación es más importante que el copyright. Pero una cooperación clandestina,

[3]Cygnus Support siguió teniendo éxito, pero luego aceptó inversiones foráneas, se volvió ambiciosa y empezó a desarrollar software no libre. Entonces fue absorbida por Red Hat, que ha lanzado la mayor parte de esos programas como software libre.

oculta no contribuye a mejorar la sociedad. Una persona debería aspirar a vivir una vida honrada abiertamente con orgullo, y esto significa decir «no» al software propietario.

Tienes derecho a poder cooperar abierta y libremente con otras personas que usan software. Tienes derecho a poder aprender cómo funciona el software, y a enseñar a tus estudiantes con él. Tienes derecho a poder contratar a tu programador favorito para arreglarlo cuando se rompa.

Tienes derecho al software libre.

Capítulo 5
¿Qué encierra un nombre?[1]

Los nombres transmiten significados; nuestra elección de los nombres determina el significado de lo que expresamos. Un nombre inadecuado dará a la gente una idea equivocada. Una rosa, llámese como se llame, destilaría el mismo olor, pero si la llamamos «lápiz», la gente podría sentirse decepcionada al intentar escribir con ella. Y si llamamos «rosas» a los lápices, la gente no entendería para qué sirven. Si llamamos «Linux» a nuestro sistema operativo, esto conduce a una idea equivocada sobre el origen del sistema, su historia y su propósito. Si lo llamamos «GNU/Linux», esto conduce —aunque no en detalle— a una idea precisa.

Pero ¿importa esto en nuestra comunidad? ¿Es importante que la gente conozca el origen del sistema, su historia y su propósito? Sí, porque quienes olvidan la historia están condenados a repetirla. El Mundo Libre que se ha desarrollado en torno a GNU/Linux no es un lugar seguro; los problemas que nos llevaron a crear GNU no han sido completamente erradicados y amenazan con volver.

Cuando explico por qué es apropiado llamar al sistema operativo «GNU/Linux» en vez de «Linux», la gente en ocasiones responde de esta manera:

> Aunque es indudable que el proyecto GNU merece el reconocimiento por esta labor, ¿merece la pena protestar si no existe tal reconocimiento? ¿No es lo importante que el trabajo se hizo, y no quién lo hizo? Usted debería relajarse, sentirse orgulloso de la labor realizada y no preocuparse por el reconocimiento.

Este podría ser un sabio consejo si esa fuera la situación —si el trabajo estuviera terminado y fuera tiempo de descansar. ¡Si tan solo eso fuera cierto! Pero los desafíos abundan y no es el momento para suponer que el futuro está garantizado. La fuerza de nuestra comunidad descansa sobre un compromiso con la libertad y la cooperación. Usar el nombre de GNU/Linux es una forma de que la gente lo recuerde e informe a los demás de nuestros objetivos.

Es posible escribir buen software libre sin pensar en GNU; muchos buenos programas se desarrollaron en nombre de Linux. Pero «Linux» se ha asociado, desde que fuera acuñado, con una filosofía que no se compromete con la libertad para cooperar. Como el nombre se emplea cada vez más en el mundo de las empresas, resultará incluso más difícil asociarlo con el espíritu comunitario.

[1]Escrito originalmente en 2000.

Un gran reto para el futuro del software libre es la tendencia de las empresas de distribución de Linux a agregar software no libre a GNU/Linux en nombre de la conveniencia y la potencia. La mayor parte de los desarrolladores de distribución comercial hacen esto; ninguno produce una distribución completamente libre. Muchos de ellos no identifican claramente los paquetes no libres de sus distribuciones. Muchos, incluso, desarrollan software no libre y lo añaden al sistema. Algunos se atreven a anunciar, de forma injuriosa, sistemas «Linux», «licenciado por puesto», lo que proporciona tanta libertad como el Windows de Microsoft.

Se justifica la inclusión de software no libre en nombre de la «popularidad de Linux» —en efecto, valoran más la popularidad que la libertad. Algunas veces se admite abiertamente. Por ejemplo, en *Wired Magazine*, Robert McMillan, editor de Linux Magazine, afirma que «el movimiento por el software de código abierto debería impulsarse sobre la base de decisiones técnicas, no políticas». Y el presidente de *Caldera* animó públicamente a los usuarios a abandonar el objetivo de la libertad y trabajar en cambio por la «popularidad de Linux».

Incluir software no libre en el sistema GNU/Linux puede aumentar su popularidad, si por popularidad entendemos el número de personas que usan GNU/Linux en combinación con software no libre. Pero al mismo tiempo se está animando implícitamente a la comunidad a aceptar el software no libre como algo positivo, y a olvidar el objetivo de la libertad. De nada sirve caminar más rápido si nos apartamos del camino.

Cuando la «adición» no libre es una biblioteca o una herramienta de programación, esto puede acabar siendo una trampa para los desarrolladores de software libre. Cuando escriben un programa que depende de un paquete no libre, su software no podrá formar parte de un sistema totalmente libre.[2]

Si nuestra comunidad sigue en esta dirección, el futuro de GNU/Linux será un mosaico de componentes libres y no libres. En el plazo de cinco años, todavía nos quedará mucho software libre, pero si nos descuidamos acabaremos por necesitar la presencia de software no libre que los usuarios esperan encontrar al lado del software libre. Si esto sucede, nuestra campaña por la libertad habrá fracasado.

Si publicar alternativas libres se redujera a una mera cuestión de programación, resolver los futuros problemas podría ser cada vez más fácil a medida que aumentaran los recursos destinados al desarrollo en nuestra comunidad. Pero algunos obstáculos amenazan con complicar las cosas: las leyes que prohíben el software libre. A medida que van acumulándose las patentes de software y que leyes como la DMCA[3] se aplican

[2]En este sentido, las bibliotecas Motif y Qt GUI entramparon enormemente al software libre en el pasado, creando problemas cuya solución llevó años. El problema de Qt está solucionado porque Qt es ahora libre; el problema de Motif no está todavía completamente resuelto, ya que su sustituto, Less Tif, necesita algo más de pulido —¡por favor voluntarios! La implementación Java no libre de Sun y las bibliotecas estándar de Java están causando ya problemas similares, reemplazarlas con software libre es uno de los mayores esfuerzos de GNU ahora.

[3]La *Digital Millenium Copyright Act* de 1998 trataba de actualizar la ley de copyright de EE.UU.; las cuestiones incluidas en la DCMA son provisiones que tienen que ver con una estratagema para la protección de los sistemas de copyright, el uso razonable y las obligaciones de los proveedores de servicios *on line*. Para más detalles sobre la DCMA véase el capítulo 12.

para prohibir el desarrollo de software libre para actividades importantes como ver un DVD o escuchar Real Audio, nos encontraremos desarmados para luchar contra los formatos de datos secretos o patentados, como no sea renunciando a los programas no libres que los usen.

Afrontar estos retos requerirá distintas clases de esfuerzo. Pero lo que necesitamos por encima de todo para confrontar cualquier desafío es recordar el objetivo de la libertad para cooperar. No podemos esperar que el mero deseo de un software potente y fiable incite a la gente a emprender grandes esfuerzos. Necesitamos la clase de determinación que la gente tiene cuando lucha por su libertad y su comunidad, determinación para seguir adelante durante años sin rendirse.

En nuestra comunidad, este principio y esta determinación emanan principalmente del proyecto GNU. Somos quienes hablamos de libertad y comunidad como algo por lo que mantenerse firmes; las organizaciones que hablan de «Linux» normalmente no aluden a esto. Las revistas sobre «Linux» suelen estar llenas de anuncios de software no libre; las empresas que empaquetan «Linux» añaden software no libre al sistema; otras «soportan Linux» con aplicaciones no libres; los grupos de usuarios de «Linux» invitan a los vendedores para presentar esas aplicaciones. El principal espacio en el que la gente de nuestra comunidad se puede aproximar más a la idea de libertad y de determinación está en el proyecto GNU.

Sin embargo, ¿cuando la gente se topa con el proyecto GNU, se siente identificada con el proyecto?

Los usuarios que son conscientes de estar utilizando un sistema originado en el proyecto GNU pueden ver una relación directa entre ellos mismos y GNU. No estarán automáticamente de acuerdo con nuestra filosofía, pero al menos tendrán una razón para pensar seriamente sobre ello. Por el contrario, quienes se consideran «usuarios de Linux» y crean que el proyecto GNU «desarrolla herramientas útiles y compatibles con Linux», perciben por lo general una relación indirecta entre GNU y ellos. Cuando entren en contacto con ella, se limitarán a ignorar la filosofía de GNU.

El proyecto GNU es idealista y cualquiera que hoy promueva el idealismo se enfrenta a un gran obstáculo: la ideología dominante anima a la gente a descartar el idealismo por ser «poco práctico». Nuestro idealismo ha sido extremadamente práctico: es la razón de que existe un sistema operativo GNU/Linux libre. La gente que disfruta de este sistema debería saber que se trata de nuestro idealismo hecho realidad.

Si el «trabajo» estuviera terminado, si no hubiera nada en juego salvo el reconocimiento, quizás sería más sabio abandonar el tema. Pero no estamos en este punto. Para inspirar a la gente a seguir trabajando, debemos obtener reconocimiento por lo que se ya ha hecho. Por favor ayúdanos llamando al sistema operativo GNU/Linux por su nombre.

Capítulo 6

Por qué «software libre» es mejor que «open source»[1]

Dado que el software libre te daría la misma libertad con cualquier otro nombre, qué nombre usemos marca una gran diferencia: palabras distintas *transmiten distintas* ideas.

En 1998, algunos dentro de la comunidad del software libre empezaron a usar el término «software *open source*»[2] en lugar de «software libre» para describir lo que hacían. El término *open source* se asoció rápidamente con un enfoque distinto, una filosofía distinta, e incluso diferentes criterios para decidir que licencias son aceptables. El movimiento de software libre y el movimiento *open source* son hoy en día movimientos separados con diferentes puntos de vista y objetivos, aunque podamos y trabajemos juntos en algunos proyectos prácticos.

La diferencia fundamental entre los dos movimientos está en sus valores, en su visión del mundo. Para el movimiento *open source*, la cuestión de si el software debe ser de fuente abierta es una cuestión práctica, no ética. Como lo expresó alguien, «el *open source* es un método de desarrollo; el software libre es un movimiento social». Para el movimiento *open source*, el software no libre es una solución ineficiente. Para el movimiento de software libre, el software no libre es un problema social y el software libre es la solución.

Relación entre el movimiento del software libre y el movimiento «open source»

El movimiento del software libre y el movimiento *open source* son como dos campos políticos dentro de la comunidad del software libre.

Grupos radicales de la década de 1960 desarrollaron una reputación de sectarismo: las organizaciones se escindían por desacuerdos en detalles estratégicos, y luego se

[1] Escrito originalmente en 1998.

[2] *Open source software* significa «software de código fuente abierto» o «software de fuente abierta». [*N. del E.*]

trataban entre sí como enemigas. O por lo menos, esa es la imagen que la gente tiene de ellos, tanto si era verdad como si no.

La relación entre el movimiento del software libre y el movimiento *open source* es justo la contraria a esa imagen. Estamos en desacuerdo en los principios básicos, pero estamos mas o menos de acuerdo en las recomendaciones prácticas. Así que trabajamos juntos en muchos proyectos específicos. No pensamos el movimiento *open source* como enemigo. El enemigo es el software propietario.

No estamos en contra del movimiento *open source*, pero no queremos que se nos mezcle. Reconocemos que han contribuido a nuestra comunidad, pero nosotros creamos esta comunidad, y queremos que la gente lo sepa. Queremos que la gente asocie nuestros logros con nuestros valores y filosofía, no con los de ellos. Queremos que se nos oiga, no estar difuminados detrás de un grupo con puntos de vista diferentes. Para prevenir que la gente piense que somos parte de ellos, hacemos malabares para no usar la palabra «open» al describir el software libre.

De modo que, por favor, menciona al movimiento del Software Libre cuando hables del trabajo que hemos hecho, y del software que hemos desarrollado —como el sistema operativo GNU/Linux.

Comparación de los dos términos

El resto de este artículo compara los términos «software libre» y *open source*. Muestra por qué el término *open source* no resuelve ningún problema, y de hecho crea algunos.

Ambigüedad

El término «software libre» conlleva un problema de ambigüedad para las personas de habla inglesa, un significado indeseado, «software que se puede obtener por un precio cero», que encaja también con el significado deseado: «software que da al usuario ciertas libertades». Nosotros hemos afrontado este problema publicando una definición más precisa de software libre, (véase «Definición de software libre») pero ésta no es una solución perfecta porque no elimina completamente el problema. Un término correcto menos ambiguo sería mejor, si no conllevara otros problemas.

Por desgracia, todas las alternativas en inglés conllevan sus propios problemas. Hemos examinado muchas alternativas que la gente ha propuesto, pero ninguna es lo suficientemente correcta como para que fuera una buena idea cambiarse a ella. Cualquier substituto para «software libre» tiene un problema semántico parecido o peor, esto incluye al «software open source».

La definición oficial de «software de código fuente abierto», tal y como está publicada por la Open Source Initiative, se acerca mucho a nuestra definición de software libre; de todos modos, es algo pobre en algunos aspectos, y han aceptado algunas licencias que nosotros consideramos inaceptablemente restrictivas para los usuarios. De todos modos, el significado obvio para «software de código fuente abierto» es que «puedes ver el código fuente». Este es un criterio más pobre que el del «software libre». «Soft-

ware de código fuente abierto» incluye software libre, pero también incluye programas semi-libres tales como Xv, e incluso algunos programas propietarios, como Qt bajo su licencia inicial (antes de la QPL).

Ese significado obvio para «fuente abierta» no es el significado que sus defensores pretenden. El resultado es que la gente a menudo malinterpreta los que esos defensores defienden. Así definió «software open source» el escritor Neil Stephenson:

> Linux es "software *open source*", lo cual quiere decir simplemente que cualquiera puede conseguir copias de sus archivos de código fuente.

No creo que él buscara deliberadamente rechazar o discutir la definición «oficial». Creo que simplemente aplicó las convenciones de la lengua inglesa para dar con una definición para el término. El estado de Kansas publicó una definición similar:

> Hagan uso del software de código abierto. El software de código abierto es software cuyo código fuente esta disponible de forma libre y pública, aunque los acuerdos específicos de licencia varían en lo que está permitido hacer con ese código.

Por supuesto, la gente del movimiento *open source* abierto han intentado afrontar esto publicando una definición precisa del término, tal y como nosotros hemos hecho con el de «software libre».

Pero la explicación de «software libre» es simple —una persona que coja la idea de «libertad de expresión, no barra libre» no volverá a equivocarse. No hay una forma breve de explicar el significado apropiado de «software open source» que muestre claramente que la definición natural es la equivocada.

Miedo a la libertad

El argumento principal del término «software open source» es que el término «software libre» hace que algunas personas se sientan incómodas. Esto es cierto: hablar sobre libertad, sobre asuntos éticos, sobre responsabilidades así como sobre conveniencia, es pedirle a la gente que piense sobre cosas que preferiría ignorar. Esto puede causar malestar, y algunas personas pueden rechazar la idea por eso. De esto no se debe deducir que la sociedad estaría mejor si dejamos de hablar de este tipo de cosas.

Años atrás, los desarrolladores de software advirtieron esta reacción de malestar y comenzaron a explorar otro enfoque para evitarlo. Se imaginaron que manteniendo el silencio sobre cuestiones de ética y de libertad, y hablando únicamente de los beneficios prácticos inmediatos de cierto software libre, podrían ser capaces de «vender» software más efectivamente a ciertos usuarios, especialmente a las empresas. El término *open source* se ofrece como una forma más de hacer esto —una forma de ser «más aceptable a las empresas». Los puntos de vista y los valores del movimiento *open source* se derivan de esta decisión.

Este enfoque ha demostrado ser efectivo en sus propios términos. Hoy mucha gente se está cambiado al software libre por razones puramente prácticas. Esto es bueno,

mientras siga ocurriendo, ¡pero eso no es todo lo que necesitamos hacer! Atraer a los usuarios al software libre no es todo el trabajo, es sólo el primer paso.

Tarde o temprano a estos usuarios se les invitará a regresar al software propietario por alguna ventaja práctica. Innumerables empresas intentan ofrecer tal tentación, y ¿por qué iban a rechazarla los usuarios? Sólo si han aprendido *a valorar la libertad* que les da el software libre, en su propio interés. Depende de nosotros difundir esta idea — y para eso tenemos que hablar de libertad. En buena parte el enfoque «guarda silencio» para acercarse a las empresas puede ser útil para la comunidad, pero también debemos tener bastante libertad de hablar.

Actualmente tenemos mucho «guarda silencio», pero no tenemos suficiente libertad de hablar. La mayoría de la gente comprometida con el software libre dice poco acerca de la libertad —normalmente porque buscan tener «más aceptación en las empresas». Los distribuidores de software muestran este patrón de forma especial. Algunos distribuidores de software GNU/Linux añaden paquetes propietarios al sistema básico libre e invitan a los usuarios a considerar esto como una ventaja, en lugar de un retroceso en la libertad.

Estamos fracasando en mantener la influencia sobre los usuarios de software libre, estamos fracasando en enseñar a la gente algo de libertad y nuestra comunidad tan pronto como entran en ella. Esta es la razón por la cual software no libre —como Qt en el momento de hacerse popular— y distribuciones de sistemas operativos parcialmente libres encuentran tierra fértil. Dejar de usar la palabra *libre* sería un error, necesitamos hablar más, no menos, de libertad.

Si los que usan el término *open source* atraen a más usuarios a nuestra comunidad, se trata en sí de una contribución, pero el resto de nosotros tendrá que trabajar más duro para llevar la cuestión de la libertad a la atención de esos usuarios. Tenemos que decir «¡es software libre y te da libertad!» más veces y más alto que nunca.

¿Podría ayudar una marca registrada?

Los defensores del *open source* trataron de convertirlo en una marca registrada, diciendo que esto podría permitirles prevenir su mal uso. Se desistió del intento cuando se dejó prescribir la solicitud en 1999, de tal forma que el rango legal del *open source* es el mismo que el del «software libre»: no hay restricción legal para su uso. He oído informaciones sobre ciertas compañías que llaman *open source* a sus paquetes de software aunque no se ajustaban a la definición oficial; yo mismo he observado algunos ejemplos.

¿Pero habría significado una gran diferencia usar un término que es una marca registrada? No necesariamente.

Las compañías también han hecho anuncios que dan la impresión de que un programa es *open source* sin decirlo explícitamente. Por ejemplo, un anuncio de IBM sobre un programa que no se ajustaba a la definición oficial decía esto: «Como es común en la comunidad *open source*, los usuarios de la tecnología [. . .] también podrán colaborar con IBM [...]»

En realidad, esto no afirma que el programa fuera *open source*, pero muchos lectores no advirtieron ese detalle. (Debería destacar que IBM estuvo tratando sinceramente de hacer de este programa software libre, y más tarde adoptaron una nueva licencia que lo hace software libre y *open source*; pero cuando se hizo el anuncio, el programa no estaba cualificado para adoptar ninguno de los dos nombres).

Y así es como Cygnus Solutions, la cual se creó para ser una compañía de software libre y más tarde se expandió, por así decirlo, al software propietario, publicitó algunos productos de software propietario: «Cygnus Solutions es un líder en el mercado de *open source* y acaba de lanzar dos productos al mercado GNU/Linux».

A diferencia de IBM, Cygnus no estaba tratando de hacer paquetes de software libre y los paquetes no se ceñían a esa calificación. Pero Cygnus realmente no dijo que esos programas fueran *open source*, ellos sólo hicieron uso de ese término para dar los lectores incautos esa impresión.

Estas observaciones nos indican que una marca registrada no hubiera impedido de verdad la confusión que viene dada con el término *open source.*

Malentendidos del «open source»

La definición de *open source* es suficientemente clara y está bastante claro que el típico programa no libre no cumple con esa calificación. Entonces pensarás que «compañía de software *open source*» podría significar aquella cuyos productos son software libre, —o que están cerca de serlo—, ¿correcto? Desafortunadamente, muchas compañías están tratando de darle un significado diferente.

En el encuentro del «Día de fabricantes de *open source*» en agosto de 1998, varios de los fabricantes comerciales invitados dijeron que ellos solamente tenían en mente convertir una parte de su trabajo en software libre —u *open source*—. El enfoque de sus negocios está en desarrollar accesorios propietarios —software o manuales— para vender a los usuarios de software libre. Nos piden que consideremos esto como legítimo, como parte de nuestra comunidad, debido a que parte del dinero se dona al desarrollo de software libre.

En efecto, estas compañías buscan ganar el distintivo favorable de *open source* para sus productos de software propietario —aún a pesar del hecho que no es software *open source*— porque tienen alguna relación con el software libre o porque la misma compañía también mantiene algo de software libre. (Un inversor de una compañía dijo de forma bastante explícita que incluirían en los paquetes libres que ellos soportan tan poco trabajo propio, como el que la comunidad les permita).

Con el paso de los años muchas compañías han contribuido al desarrollo del software libre. Algunas de estas compañías desarrollaban principalmente software no libre, pero las dos actividades estaban separadas; así, podíamos ignorar sus productos no libres, y trabajar con ellos en proyectos de software libre. Así que tiempo después podríamos estarles sinceramente agradecidos por sus contribuciones al software libre, sin tener que hablar sobre el resto de lo que hacían.

No podemos hacer lo mismo con estas nuevas compañías, porque no nos lo van a permitir. Estas compañías tratan activamente de conducir al público a mezclar todas sus actividades, quieren que estimemos su software no libre tan favorablemente como lo haríamos con una auténtica contribución, a pesar de que la suya no lo sea. Se presentan a sí mismas como «compañías *open source*», esperando que desarrollemos un sentimiento cálido y tierno hacia ellas, y que estemos locos por demostrarlo.

Esta práctica manipuladora no podría ser menos dañina si se hiciera usando el término «software libre». Pero parece que las compañías no usan el término «software libre» de esa manera; tal vez su asociación con el idealismo lo hace parecer inapropiado. El término *open source* abrió la puerta a esta práctica.

En una feria sectorial a finales de 1998, dedicada al sistema operativo a menudo denominado «Linux», el conferenciante invitado era un ejecutivo de una prominente compañía de software. Probablemente lo invitaron teniendo en cuenta la decisión de su compañía de «apoyar» ese sistema. Desafortunadamente, su forma de «apoyo» consiste en sacar al mercado software no libre compatible con el sistema —en otras palabras, usando nuestra comunidad como un mercado pero sin contribuir a ella.

Dijo, «no hay forma de que nosotros hagamos *open source* nuestro producto, pero quizá lo hagamos *open source* internamente. Si permitimos que nuestro personal de asistencia al cliente tenga acceso al código fuente, podrían corregir errores para los clientes y podríamos proporcionar un mejor producto y un mejor servicio». (Esto no es una cita exacta, porque no tomé nota de sus palabras, pero refleja la idea clave).

Personas del público me dijeron después, «es que no entiende la cuestión». Pero, ¿es eso cierto? ¿Qué cuestión no entendió?

A él no se le escapó la esencia del movimiento *open source*. Dicho movimiento no dice que lo usuarios deban tener libertad, únicamente dice que permite a más gente acceder al código fuente y al ayudar a mejorarlo conduce a un mejor y más rápido desarrollo. El ejecutivo comprendió esa cuestión completamente; no deseoso de desarrollar ese enfoque al completo, incluyendo a los usuarios, estaba pensando en implementarlo parcialmente, dentro de la compañía.

La cuestión que se le escapó es la cuestión para cuya discusión no está diseñado el *open source*: la cuestión de que los usuarios tienen derecho a la libertad.

Difundir la idea de libertad es un gran trabajo —necesita de tu ayuda. Por eso en el proyecto GNU somos fieles al término «software libre», y así poder ayudar a hacer ese trabajo. Si te parece que la libertad y la comunidad son importantes en sí mismas —no sólo por las ventajas que conllevan— por favor únete a nosotros usando el término «software libre».

Capítulo 7

Cómo promover el software libre si trabajas en la Universidad[1]

En el movimiento del software libre creemos que los usuarios de ordenadores deberían tener libertad para cambiar y redistribuir el software que utilizan. El adjetivo «libre» en el software libre hace referencia a la libertad: libertad del usuario para ejecutar, modificar y redistribuir software. El software libre contribuye al saber humano, al contrario que el software propietario. Por este motivo, las universidades deberían fomentar el software libre, para hacer una aportación al progreso del conocimiento humano, del mismo modo que deben animar a científicos y académicos a publicar sus obras.

Pero el software (y la ciencia) despiertan la codicia en un gran número de gerentes universitarios: consideran los programas como una potencial fuente de ingresos, y no como aportaciones al saber humano. Los programadores de software libre llevan conviviendo con esta tendencia desde hace casi veinte años.

Cuando comencé a desarrollar el sistema operativo GNU en 1984, lo primero que hice fue renunciar a mi trabajo en el MIT. Hice esto precisamente para que así la oficina de licencias del MIT fuera incapaz de interferir en la publicación de GNU como software libre. Había diseñado una estrategia para licenciar los programas contenidos en GNU que garantizaría que todas las versiones modificadas seguirían siendo software libre, una estrategia que acabaría convirtiéndose en la GNU General Public License (GNU GPL).[2] No quería tener que rogarle a la administración del MIT que me permitiera usarla.

Con el paso de los años, varias filiales universitarias han acudido con frecuencia a la Free Software Foundation para asesorarse sobre la forma de negociar con los gerentes que opinan que el software es tan sólo algo que vender. Un buen método, aplicable incluso a proyectos específicamente financiados, consiste en basar su trabajo en un programa ya existente publicado con GNU GPL. De esta forma, se puede responder a los gerentes: «No podemos publicar la versión modificada, a menos que sea con GNU GPL, de otro modo estaríamos infringiendo el copyright». Una vez desaparecido cualquier

[1] Escrito originalmente en 2002.

[2] En castellano, Licencia Pública General GNU.

rastro del símbolo del dólar de sus ojos, por lo general consentirán en publicarlo como software libre.

También se puede pedir ayuda al patrocinador del proyecto. Cuando un equipo de la NYU (Universidad de Nueva York) desarrolló el compilador GNU Ada con fondos procedentes de las Fuerzas Aéreas de los EE.UU., el contrato especificaba que el código resultante se donaría a la Free Software Foundation. Primero se negocia el acuerdo con el patrocinador, luego se explica cortésmente a la administración de la universidad que no habrá renegociación de ninguna clase. Dado que la administración prefiere tener un contrato para desarrollar software libre antes que quedarse con las manos vacías, lo más probable es que acepten el trato.

Hagáis lo que hagáis, habrá que plantear la cuestión cuanto antes —desde luego, antes de que el programa esté a medio camino. Llegados este punto, la universidad todavía os necesita, así que podréis jugar duro: advertir a la administración de que el programa se terminará y se dejará listo para ser usado, siempre y cuando acuerden por escrito convertirlo en software libre —y acepten la licencia de software libre de vuestra elección. De lo contrario, sólo alcanzaréis a escribir una ponencia al respecto y nunca desarrollaréis una versión lo bastante buena para publicarse. Cuando los gestores comprendan que sus opciones se limitan a tener un paquete de software libre que aportará prestigio a la universidad o nada de nada, por lo general se decantarán por la primera opción.

No todas las universidades tienen políticas codiciosas. La política de la Universidad de Texas tiene una política que facilita que todo el software desarrollado en ella se publique como software libre bajo la licencia GPL. Univates en Brasil y el Indian Institute of Information Technology en Hyberabad, India, practican políticas de publicación de software con GPL. Si os ganáis primero el apoyo del profesorado, es posible que logréis instituir una política semejante en vuestra universidad. Exponedlo como una cuestión de principio: ¿tiene la universidad la misión de contribuir al progreso del saber humano o su único objetivo es perpetuarse a sí misma?

Sea cual sea vuestra postura, siempre conviene mostrar determinación y adoptar una perspectiva ética, tal y como lo hacemos nosotros en el movimiento del software libre. Si deseamos tratar al público éticamente, el software debería ser libre en el sentido de libertad para todos.[3]

Muchos programadores de software libre se limitan a alegar razones prácticas: defienden que el software se comparta con otros y se modifique como forma de acelerar la creación de un software potente y fiable. Si son estos valores los que os mueven a desarrollar software libre, muy bien, se agradece vuestra contribución. Pero no son la clase de valores que os permitirán hacer frente con firmeza a los gestores de la universidad que pretendan hacer propietario vuestro programa.

Por ejemplo, pueden aducir: «podríamos hacer mas potente y fiable el programa con todo el dinero que se obtenga de las ventas». Esto puede resultar cierto o no, pero a priori es difícil presentar argumentos en contra. Pueden sugerir una licencia que ofrezca copias «gratuitas, sólo para uso académico», y así lanzar el mensaje al público de que

[3] Algo obvio en castellano, pero no en inglés donde *free* significa también «gratis».

no merecen esa libertad y además alegar que de esta manera lograrías la cooperación de la comunidad académica, que es —razonarían— todo lo que necesitas.

Si se parte de unos valores «pragmáticos», es difícil plantear buenas razones para rechazar estas propuestas, que son en realidad un callejón sin salida, sucede lo contrario al fundamentar nuestra postura en valores éticos y políticos. ¿De qué serviría un programa potente y fiable a costa de la libertad de los usuarios? ¿No debería la libertad estar presente tanto fuera como dentro de la academia? Las respuestas resultan obvias siempre que la libertad y el bien de la comunidad figuren entre nuestros objetivos. El software libre respeta la libertad de los usuarios, mientras que el software propietario la niega.

Nada mejor que ser consciente, en este caso en concreto, para aumentar nuestra determinación de que la libertad de la comunidad depende de nosotros.

Capítulo 8

Vender software libre[1]

Mucha gente cree que el espíritu del proyecto GNU consiste en que no se debe poner precio a la distribución de copias de software, o que se debe cobrar lo menos posible —lo suficiente para cubrir costes.

En realidad, nosotros defendemos que quienes redistribuyen software libre cobren cuanto quieran o cuanto puedan. Si esto te sorprende, sigue leyendo, por favor.

La palabra «libre» admite dos significados generales; se puede referir tanto a la libertad como al precio. Cuando hablamos de «software libre» nos referimos a la libertad y no al precio. En concreto, esto significa que un usuario es libre de instalar un programa, cambiarlo y redistribuirlo con o sin cambios.

Algunas veces los programas libres se distribuyen gratis y otras a cambio de un precio cuantioso. A menudo un mismo programa está disponible de ambas formas en diferentes lugares. El programa es libre sin que importe su precio, porque los usuarios lo pueden utilizar libremente.

Los programas que no son libres se venden frecuentemente a precios altos, pero a veces una tienda te ofrece una copia sin cobrarte. No obstante, eso no lo convierte en software libre. Con o sin precio, el programa no es libre porque los usuarios no tienen libertad.

Dado que el software libre no es una cuestión de precio, un precio bajo no resulta más libre, ni más próximo a lo libre. De modo que si redistribuyes copias de software libre, también puedes fijar un precio cuantioso para hacer algo de dinero. Redistribuir software es una actividad buena y legítima; si lo haces, también puedes sacar una ganancia por ello.

El software libre es un proyecto comunitario y cualquiera que lo integre debería buscar formas de contribuir a la construcción de la comunidad. La manera de conseguir esto, para un distribuidor, consiste en donar una parte de los beneficios a la Free Software Foundation o a cualquier otro proyecto que desarrolle el software libre. Financiando el desarrollo harás avanzar el mundo del software libre.

¡Distribuir software libre es una oportunidad de obtener medios para el desarrollo! ¡No la pierdas!

Para aportar fondos, necesitas tener alguna ganancia. Si fijas precios demasiado bajos, no te quedará nada para apoyar el desarrollo del software libre.

[1] Escrito originalmente en 1996

¿Puede perjudicar un precio de distribución más alto a algunos usuarios?

A la gente a veces le preocupa que un alto precio de distribución ponga al software libre fuera del alcance de usuarios sin mucho dinero. Con el software propietario, un precio alto provoca exactamente esto —pero el software libre es diferente.

La diferencia es que el software libre tiende a difundirse de forma natural y hay muchas formas de obtenerlo.

Los monopolistas del ámbito del software intentan como locos que no instales un programa propietario sin pagar el precio de mercado. Si dicho precio es alto, hará difícil que muchos usuarios utilicen ese programa.

Con el software libre, los usuarios no tienen que pagar un precio de distribución para utilizar software. Pueden copiar el programa de un amigo que tenga una copia o con la ayuda de un amigo que tenga acceso a la red. O se pueden juntar varios usuarios, pagar a escote un CD ROM e instalarse el software por turnos, que el CD ROM tenga un precio alto no supone un gran impedimento cuando el software es libre.

¿Puede desalentar un precio de distribución más alto el uso de software libre?

Otro motivo de preocupación común está relacionado con la popularidad del software libre. La gente piensa que los precios de distribución altos podrían reducir el número de usuarios, o que los precios bajos pueden alentarlo.

Esto resulta cierto para el software propietario —pero el software libre es distinto. Con tantas maneras de conseguir copias, el precio del servicio de distribución tiene menos efecto en su popularidad.

A largo plazo, la cantidad de gente que use software libre depende principalmente de cuanto puede hacer con el software libre y de lo fácil que sea su manejo. Muchos usuarios seguirán utilizando software propietario si el software libre no puede hacer todos los trabajos que ellos desean. Así, si queremos incrementar el número de usuarios a largo plazo, deberemos sobre todo desarrollar más software libre.

La vía más directa para hacer esto es escribir por tu cuenta el software libre y los manuales necesarios. Pero si te dedicas a la distribución antes que a la programación, el mejor modo de ayudar es conseguir fondos para que otros lo escriban.

La expresión «vender software» también puede ser confusa

Rigurosamente «vender» significa intercambiar bienes por dinero. Vender una copia de un programa libre es legítimo y nosotros lo alentamos.

De todos modos, cuando la gente piensa acerca de «vender software», normalmente imagina que se hace del mismo modo que la mayoría de las empresas: produciendo software propietario mejor que software libre.

Por lo tanto, a menos que se establezcan las distinciones con cuidado, como se hace en este artículo, aconsejamos que se evite la expresión «vender software» y que se elija otra fórmula en su lugar. Por ejemplo, se podría decir «distribuir software libre por un precio» —una fórmula no ambigua.

Altos o bajos precios y la GNU GPL

Excepto en un caso especial, la licencia GPL (General Public License) de GNU no establece requisitos sobre cuánto puedes cobrar por distribuir una copia de software libre. Puedes no cobrar nada, un penique, un dólar o un billón de dólares. Depende de ti y del mercado, así que no te nos quejes si nadie quiere pagar un billón de dólares por una copia.

La única excepción se da en el caso de que los binarios se distribuyan sin su código fuente completo. A los que hacen esto, la GNU GPL les obliga a proporcionar en lo sucesivo el código fuente. Sin un límite en el precio para el código fuente, serían capaces de fijar una cantidad demasiado grande como para que alguien la pague —como un billón de dólares, por ejemplo— y de este modo fingir que el código fuente se está publicando, cuando en realidad se está ocultando. Por lo tanto en este caso tenemos que limitar el precio de la fuente para asegurar la libertad del usuario. En situaciones normales, de todos modos, no existe tal justificación para limitar los precios de distribución, así que no los limitamos.

A veces las compañías cuyas actividades traspasan la línea de lo que la GNU GPL permite, suplican un permiso y dicen «no cobraremos por el software GNU», o algo parecido. Así no llegan a ninguna parte. El software libre tiene que ver con la libertad y reforzar la GPL es defender la libertad. Al defender la libertad de los usuarios, no nos distraen cuestiones secundarias como cuánto se cobra por una distribución. La libertad es la cuestión, toda la cuestión y la única cuestión.

Capítulo 9

El software libre necesita documentación libre[1]

La mayor deficiencia en los sistemas operativos libres no se encuentra en el software, sino en la falta de buenos manuales libres que podamos incluir en esos sistemas. Muchos de nuestros programas más importantes no vienen acompañados de manuales completos. La documentación es una parte esencial de cualquier paquete de software; cuando un paquete de software libre relevante no está acompañado de un manual libre, se da una tremenda laguna. Hoy en día tenemos muchas de estas lagunas.

Érase una vez, hace muchos años, pensé «voy a aprender Perl». Conseguí una copia de un manual libre, pero lo encontré difícil de leer. Cuando pedí a los usuarios de Perl que me dieran alternativas, me dijeron que había mejores manuales de introducción, pero estos no eran libres.

¿Qué ocurría? Los autores habían escrito buenos manuales para O´Reilly Associates, que los editó con condiciones restrictivas —no se podían copiar ni modificar, y los archivos originales no estaban disponibles—, lo que los dejaba al margen de la comunidad del software libre.

No era la primera vez que había pasado tal cosa y —para desgracia de nuestra comunidad— estaba lejos de ser la última. Desde entonces, los editores de manuales propietarios han incitado a muchísimos autores a hacer restrictivos sus manuales. Cuántas veces habré oído a un usuario de GNU hablarme apasionadamente sobre un manual que está escribiendo, con el que espera ayudar al proyecto GNU y después me deja con un palmo de narices, mientras procede a explicar que ha firmado un contrato con un editor que lo limitará tanto que no podremos usarlo.

Dado que entre los programadores escribir bien en un inglés correcto es una habilidad poco habitual, difícilmente nos podremos permitir perder manuales de esta manera.

La documentación libre, como el software libre, es un asunto de libertad y no de precio. El problema con estos manuales no era que O´Reilly pusiera un precio por ejemplar impreso —lo cual en sí está bien. (La Free Software Foundation también vende ejemplares impresos de manuales sobre GNU). Pero los manuales de GNU están disponibles con su código fuente, mientras que estos manuales sólo están disponibles en papel. Los

[1] Escrito originalmente en 2000.

manuales de GNU vienen con un permiso de copia y modificación; los manuales de Perl, no. Estas restricciones son el problema.

El criterio con los manuales libres es bastante parecido al del software libre: se trata de proporcionar ciertas libertades a todos los usuarios. La distribución —incluyendo la distribución comercial— debe ser permitida, de modo que el manual pueda acompañar a cada copia del programa, en papel o en la Red. El permiso de modificación es también crucial.

Como norma general, no creo que tener permiso para modificar todo tipo de artículos y libros resulte esencial para la gente. Los problemas para el texto impreso no son necesariamente los mismos que los del software. Por ejemplo, no me parece que tú o yo estemos obligados a dar permiso para modificar artículos como éste, que describen nuestras prácticas y nuestros puntos de vista.

Pero hay un motivo particular por el que la libertad de modificación es crucial para la información que acompaña al software libre. Cuando la gente ejercita su derecho a modificar el software y añade o cambia sus características, si es concienzuda también cambiará su correspondiente manual —de este modo pueden suministrar información precisa y útil con el programa modificado. Un manual que impide a los programadores ser concienzudos y acabar el trabajo, o para ser más exactos, que les obliga a escribir desde cero un nuevo manual si cambian el programa, no responde a las necesidades de nuestra comunidad.

Si bien es inaceptable prohibir de pleno la modificación, cierto tipo de límites a los medios de modificación no suponen ningún problema. Por ejemplo, son aceptables las exigencias de mantener la nota del *copyright* original del autor, las condiciones de distribución o la lista de autores. Tampoco es un problema obligar a que en las versiones modificadas aparezca constancia de que han sido modificadas, o incluso tener secciones enteras que no pueden ser borradas o cambiadas, siempre que esas secciones traten sobre asuntos no técnicos. (Algunos manuales de GNU las tienen).

Este tipo de restricciones no son un problema porque, en la práctica, no impiden que el programador concienzudo adapte el manual para que corresponda con el programa modificado. En otras palabras, no coartan a la comunidad del software libre en su pleno uso del manual.

De todos modos, debe ser posible modificar todo el contenido técnico del manual y luego distribuir el resultado a través de todos los medios, a través de todos los canales habituales; de no ser así, las restricciones coartarán a la comunidad, el manual no será libre y por lo tanto necesitaremos otro.

Por desagracia, a menudo cuesta encontrar a alguien que escriba otro manual cuando ya existe un manual propietario. El obstáculo es que muchos usuarios piensan que un manual propietario resulta suficientemente aceptable —y de este modo no consideran la necesidad de escribir un manual libre. No comprenden que el sistema operativo libre tiene una necesidad que se debe cubrir.

¿Por qué piensan los usuarios que los manuales propietarios son suficientes? Algunos no se han parado a pensar en ello. Espero que este artículo ayude a cambiar esta situación.

Otros usuarios consideran aceptables los manuales propietarios por el mismo motivo que mucha gente considera aceptable el software propietario: juzgan según términos puramente prácticos, sin considerar la libertad como criterio. Esta gente tiene derecho de opinar así, pero dado que estas opiniones brotan de valores que no incluyen la libertad, no son una guía para los que sí valoramos la libertad.
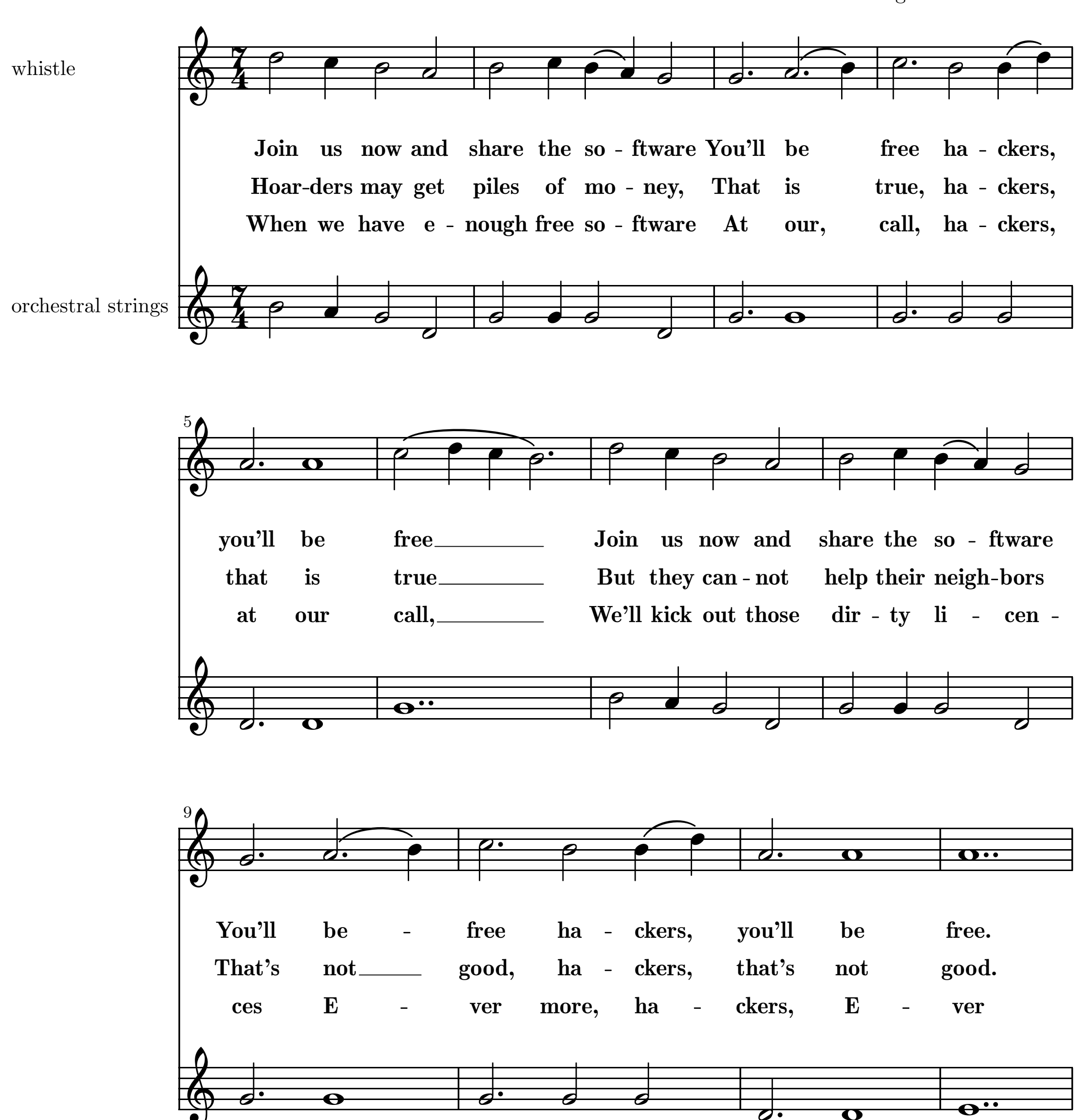
Por favor, difunde estas idea sobre este asunto. Seguimos perdiendo manuales en provecho de las ediciones propietarias. Si difundimos la idea de que los manuales propietarios no son suficientes, quizá el próximo que quiera ayudar a GNU escribiendo una guía se dará cuenta, antes de que sea demasiado tarde, de que por encima de todo debe ser libre.

También podemos animar a los editores comerciales a vender manuales libres basados en copyleft en lugar de manuales propietarios. Puedes ayudar si compruebas las condiciones de distribución de un manual antes de comprarlo y eliges los manuales copyleft frente los que no los son.

(Nota: La Free Software Foundation mantiene la página web `http://www.gnu.org/doc/other-free-books.html`, que tiene un listado de libros copyleft disponibles a través de otros editores.)

Capítulo 10
La canción del software libre

La melodía es la de la canción popular búlgara “Sadi Moma”.



Parte II

Copyright, copyleft, patentes

Capítulo 11
El derecho a leer[1]

(De «El camino a Tycho», una colección de artículos sobre los antecedentes de la *Revolución Lunaria*, publicado en *Luna City* en 2096.)

Para Dan Halbert, el camino hacia Tycho comenzó en la universidad, cuando Lissa Lenz le pidió prestado su ordenador. El suyo se había estropeado, y a menos que pudiese usar otro suspendería el proyecto de fin de trimestre. Ella no se habría atrevido a pedírselo a nadie, excepto a Dan.

Esto puso a Dan en un dilema. Tenía que ayudarla, pero si le prestaba su ordenador ella podría leer sus libros. Dejando a un lado el peligro de acabar en la cárcel durante muchos años por permitir a otra persona leer sus libros, al principio la simple idea le sorprendió. Como todo el mundo, había aprendido desde los años de colegio que compartir libros era malo, algo que sólo un pirata haría.

Además, era muy improbable que la SPA —Software Protection Authority, [Autoridad para la Protección del Software]— lo descubriese. En sus clases de programación, había aprendido que cada libro tenía un control de copyright que informaba directamente a la oficina central de licencias de cuándo y dónde se estaba leyendo, y quién leía —utilizaban esta información para descubrir a los piratas de la lectura, pero también para vender perfiles personales a otros comercios. La próxima vez que su ordenador se conectase a la red, la oficina central de licencias lo descubriría todo. Él, como propietario del ordenador, recibiría el castigo más duro por no tomar las medidas necesarias para evitar el delito.

Por supuesto, podría ser que Lissa no quisiera leer sus libros. Probablemente lo único que necesitaba del ordenador era redactar su proyecto. Pero Dan sabía que ella provenía de una familia de clase media, que a duras penas se podía permitir pagar la matrícula y no digamos las tasas de lectura. Leer sus libros podía ser la única forma por la que podría terminar la carrera. Comprendía la situación; él mismo había pedido un préstamo para pagar por los artículos de investigación que leía —el 10 % de ese dinero iba a parar a sus autores y como Dan pretendía hacer carrera en la Universidad, esperaba que sus artículos de investigación, en caso de ser citados frecuentemente, le darían suficientes beneficios como para pagar el crédito.

[1]Escrito originalmente en el número de febrero de 1997 de la revista *Communications of the ACM* (Volumen 40, Número 2). La «Nota del Autor» fue actualizada en 2002.

Con el paso del tiempo, Dan descubrió que hubo una época en que todo el mundo podía acudir a una biblioteca y leer artículos, incluso libros, sin tener que pagar. Había investigadores independientes que podían leer miles de páginas sin necesidad de recurrir a becas de biblioteca. Pero desde los años noventa del siglo anterior, las editoriales, tanto comerciales como no comerciales, habían empezado a cobrar por el acceso a los artículos. En 2047, las bibliotecas con acceso público a literatura académica eran sólo un vago recuerdo.

Había formas de saltarse los controles de la SPA y de la oficina central de licencias. Pero también eran ilegales. Dan conoció a un compañero de clase, Frank Martucci, que consiguió una herramienta ilegal de depuración y la usaba para saltarse el control de *copyright* de los libros. Pero se lo contó a demasiados amigos, y uno de ellos le denunció a la SPA a cambio de una recompensa —era fácil tentar a los estudiantes endeudados para traicionar a sus amigos. En 2047, Frank estaba en la cárcel, pero no por pirateo, sino por tener un *depurador*.

Dan averiguó más tarde que hubo un tiempo en que cualquiera podía tener un depurador. Había incluso depuradores gratuitos en CD o disponibles libremente en la red. Pero los usuarios normales empezaron a usarlos para saltarse los controles de *copyright* y por fin un juez dictaminó que ése se había convertido en su principal uso práctico. Eso significaba que los depuradores eran ilegales y los programadores que los crearon fueron a parar a la cárcel.

Obviamente, los programadores aún necesitan depuradores, pero en 2047 sólo había copias numeradas de los depuradores comerciales, y sólo estaban disponibles para los programadores oficialmente autorizados. El depurador que Dan había utilizado en sus clases de programación estaba detrás de un cortafuegos para que sólo pudiese utilizarse en los ejercicios de clase.

También se podía saltar el control de *copyright* instalando el kernel de un sistema modificado. Dan descubrió que hacia el cambio de siglo hubo kernels libres, incluso sistemas operativos completos. Pero ahora no sólo eran ilegales, como los depuradores. No se podía instalar sin saber la clave de superusuario del ordenador y ni el FBI ni el servicio técnico de Microsoft la revelarían.

Dan llegó a la conclusión de que simplemente no podía dejarle a Lissa su ordenador. Pero no podía negarse a ayudarla, porque estaba enamorado de ella. Cada oportunidad de hablar con ella era algo maravilloso. Y el hecho de que le hubiese pedido ayuda a él podía significar que ella sentía lo mismo.

Dan resolvió el dilema haciendo algo incluso más increíble, le dejó su ordenador y le dio su clave. De esta forma, si Lissa leía sus libros, la oficina central de licencias pensaría que era él quien estaba leyendo. Seguía siendo un delito, pero la SPA no lo detectaría automáticamente. Sólo podrían descubrirlo si Lissa le denunciaba.

Si la universidad descubriese que le había dado su clave a Lissa, significaría la expulsión de ambos, independientemente del uso que hubiera hecho ella de su clave. La política de la Universidad era que cualquier interferencia con sus métodos de control sobre el uso de los ordenadores era motivo de acción disciplinaria. No importaba el

daño, el delito era el hecho de dificultar el control. Se daba por supuesto que esto significaba que se estaba haciendo algo prohibido, no necesitaban saber qué.

En realidad, los estudiantes no eran expulsados, no directamente. En lugar de eso, se les prohibía el acceso a los ordenadores de la universidad, lo que equivalía a suspender sus asignaturas.

Dan supo más tarde que ese tipo de políticas en la Universidad comenzó durante la década de 1980, cuando los estudiantes empezaron a usar los ordenadores en masa. Antes, las universidades tenían una actitud diferente: sólo se penalizaban las actividades peligrosas, no las meramente sospechosas.

Lissa no denunció a Dan a la SPA. Su decisión de ayudarla llevó a que se casaran y también a que cuestionaran lo que les habían enseñado cuando eran niños sobre la piratería. Empezaron a leer sobre la historia del *copyright*, sobre la Unión Soviética y sus restricciones sobre las copias, e incluso sobre la constitución original de los Estados Unidos. Se mudaron a Luna City, donde se encontraron con otros que intentaban librarse del largo brazo de la SPA de la misma manera. Cuando el Levantamiento de Tycho se produjo en 2062, el derecho universal a leer se convirtió en uno de sus objetivos fundamentales.

Nota del autor

El derecho a leer es una batalla que se está librando hoy en día. Aunque nuestra forma de vida actual podría tardar cincuenta años en desaparecer, la mayoría de las leyes y de las prácticas descritas anteriormente ya han sido propuestas, y muchas han entrado en vigor dentro y fuera de los Estados Unidos. En EE.UU., el *Digital Millenium Copyright Act* de 1998 estableció la base legal para restringir la lectura y el préstamo de libros informatizados —así como de otras clases de datos. La Unión Europea impuso restricciones similares con su directiva sobre *copyright* de 2001.

Hasta hace poco había una excepción, la idea de que el FBI y Microsoft guardaran las claves de administración de los ordenadores personales y no las dejasen tener no fue propuesta hasta 2002: se le denomina «Informática de Confianza» o «Palladium».

Cada vez estamos más cerca de este punto. En 2001 el Senador Hollings, financiado por la Disney, propuso un proyecto de ley llamado SSSCA —ahora rebautizado como la CBDTPA— que podría requerir que todos los nuevos ordenadores tuviesen aplicaciones obligatorias de restricción de copia que el usuario no podría puentear

En 2001 los Estados Unidos, empezaron a intentar utilizar la llamada Área de Libre Comercio de las Américas (ALCA) para imponer las mismas normas en todos los países del hemisferio occidental. El ALCA es uno de los denominados tratados de «libre comercio», dirigido actualmente a otorgar mayor poder a las empresas sobre los gobiernos democráticos; imponiendo leyes como la DMCA que son típicas de su espíritu. La Electronic Frontier Foundation anima a la gente a que explique a sus gobiernos por que deberían oponerse a esos planes.

La SPA que en realidad corresponde a la Software Publisher's Association, ha sido reemplazada en su papel por la BSA o Business Software Aliance. La BSA no es un

cuerpo de policía oficial. Actúa como tal extraoficialmente. Usa métodos de delación que tienen reminiscencias en la antigua Unión Soviética. Anima a la gente a informar sobre sus compañeros de trabajo y sus amigos. Promovió una campaña de terror en Argentina, durante 2001, amenazando con la cárcel a todo aquel que compartiese software.

Cuando este artículo fue escrito, la SPA amenazaba a los pequeños proveedores de Internet para que le permitiesen controlar a todos sus usuarios. Muchos ISP cedieron ante las amenazas, ya que no podían permitirse recurrir a la vía judicial. Al menos un ISP, Community ConneXion, en Oakland (California), se negó a ceder a las presiones y ha sido demandado. Aparentemente, la SPA retiró la demanda hace poco, pero no hay duda de que continuarán su campaña por otros medios.

Las políticas universitarias de seguridad descritas arriba no son imaginarias. Por ejemplo, el ordenador de una universidad de la zona de Chicago despliega el siguiente mensaje al entrar en el sistema:

> «Este sistema sólo puede ser utilizado por usuarios autorizados. Cualquier persona que utilice este sistema sin autorización o fuera de los límites autorizados será vigilado por el personal administrador del sistema. Durante el control de usuarios que realicen actividades no autorizadas o durante el mantenimiento del sistema, las actividades de los usuarios autorizados podrán ser supervisadas. Cualquiera que utilice este sistema acepta expresamente este control y deberá saber que, en caso de que dicho control revelara posibles indicios de actividades ilegales o de violación de las normas de la universidad, el personal de mantenimiento del sistema podrá proporcionar estas pruebas a las autoridades de la Universidad y/o a las fuerzas de seguridad.»

Esta es una interesante interpretación de la Cuarta Enmienda: obligar a los usuarios a renunciar por adelantado a los derechos contemplados en ella.

Referencias

- El *White Paper* del Gobierno*: Information Infraestructure Task Force*, *Intellectual Property and the National Information Infraestructure: The Report of the Working Group on Intellectual Property Rights* (1995).
- *An explanation of the White Paper: The Copyright Grab*, Pamela Samuelson, Wired, enero de 1996.
- *Sold Out*, James Boyle, The New York Times, 31 de marzo de 1996.
- *Public Data or Private Data*, The Washington Post, 4 de noviembre de 1996.
- *Union for the Public Domain*, una nueva organización que pretende resistirse y frenar la desmedida generalización de la propiedad intelectual.

Capítulo 12
Malinterpretar el copyright: una sucesión de errores[1]

Algo extraño y peligroso está sucediendo en la legislación sobre el copyright. Según la Constitución de los Estados Unidos, el copyright existe para beneficiar a los usuarios —los que leen libros, escuchan música o utilizan software—, no para beneficiar a los editores o los autores. Aún cuando la gente tiende cada vez más a rechazar y desobedecer las limitaciones del copyright que le vienen impuestas «por su propio beneficio», el gobierno de los Estados Unidos está añadiendo más restricciones y trata de asustar al público con nuevas y rígidas penas con el fin de conducirlo a la sumisión. ¿Cómo llegó la política de copyright a ser diametralmente opuesta a su propósito establecido? ¿Cómo podemos volver a alinearla con ese propósito? Para comprenderlo, debemos empezar por considerar la raíz de la legislación estadounidense sobre copyright: la Constitución de los Estados Unidos.

El copyright en la Constitución de los Estados Unidos

Cuando se diseñó la Constitución de los Estados Unidos, se propuso la idea de que se concediera a los autores un monopolio sobre el copyright —la propuesta fue rechazada. Los fundadores de nuestro país adoptaron un supuesto diferente: que el copyright no es un derecho natural de los autores, sino una concesión artificial que se les hace en nombre del progreso. La Constitución concede carta legal al sistema de copyright con este párrafo (Artículo I, Sección 8):

> [El Congreso tendrá el poder] de promover el progreso de la ciencia y las artes provechosas, asegurando por tiempo limitado a los inventores y autores el exclusivo derecho sobre sus respectivos descubrimientos y escritos.

El Tribunal Supremo ha afirmado repetidas veces que promover el progreso significa un beneficio para los usuarios de las obras sujetas a copyright. Por ejemplo, en el caso de la Fox Film contra Doyal, el tribunal dictó:

[1]Publicado originalmente en este libro.

> El exclusivo interés de los Estados Unidos y el objeto primordial de conceder el monopolio [del copyright] reside en los beneficios generales obtenidos por el público a partir del trabajo de los autores.

Esta decisión fundamental explica por qué el copyright no es *obligatorio* según la Constitución y es sólo algo *permitido* como una opción —también explica por qué se supone que su duración alcanza un «tiempo limitado». Si el copyright fuera un derecho natural, algo que los autores tienen en tanto depositarios de ese derecho, nada justificaría la extinción de este derecho pasado un determinado periodo de tiempo, como si los hogares de cada uno pasaran a ser propiedad pública cierto tiempo después de su construcción.

El «contrato[2] de copyright»

El sistema de *copyright* funciona mediante la concesión de privilegios y por lo tanto de beneficios, a los editores y a los autores, pero no lo hace en su provecho. Más bien lo hace para modificar su comportamiento: proporciona un incentivo a los autores para escribir y editar más. En la práctica, el gobierno emplea los derechos naturales del público, en nombre del público, como parte de un trato para ofrecerle un mayor número de obras editadas. Los expertos en derecho llaman a este concepto el «contrato de copyright»; como la adquisición estatal de una autopista o un avión usando el dinero de los contribuyentes, excepto que en este caso el gobierno gasta nuestra libertad en lugar de nuestro dinero.

Pero ¿tal y como existe en la actualidad, el contrato supone un buen trato para el público? Son posibles muchos contratos alternativos; ¿cuál es el mejor? Cualquier medida en relación a la política de copyright es parte de esta cuestión. Si malinterpretamos la naturaleza de la cuestión, tenderemos a tomar medidas de forma errónea.

La Constitución permite que se concedan derechos de copyright a los autores. En la práctica, normalmente los autores se los ceden a los editores; son los editores, y no los autores, quienes suelen ejercer los derechos y quienes se quedan con la mayoría de los beneficios, aunque los autores consigan una pequeña porción. Por eso los editores son los que con frecuencia presionan más para aumentar los poderes del copyright. Para reflejar mejor la realidad del copyright en lugar del mito, este artículo se refiere a los editores antes que a los autores como sujetos de los derechos del copyright. También se refiere a los usuarios de las obras protegidas por el copyright como «lectores», aún cuando este uso no siempre signifique su lectura, en la medida en que el término «usuario» resulta lejano y abstracto.

El primer error: «equilibrar la balanza»

El contrato de copyright pone al público en primer término: el beneficio para el público lector es un fin en sí mismo; los beneficios para los editores —si es que se dan—

[2]Nótese que el termino inglés *bargain* puede referirse tanto a un trato como a un chollo. [*N. del E.*]

son sólo medios para conseguir ese fin. En principio, los intereses de los lectores y los de los editores son cualitativamente desiguales. El primer paso al malinterpretar el propósito del copyright es elevar a los editores al mismo nivel de importancia que a los lectores.

Se ha dicho con frecuencia que la legislación estadounidense de copyright se propone equilibrar la balanza entre los intereses de los editores y de los lectores. Los que citan esta interpretación la presentan como una reformulación del punto de partida establecido en la Constitución; en otras palabras, se la supone equivalente al contrato de copyright.

Pero las dos interpretaciones están lejos de ser equivalentes; son conceptualmente distintas y sus implicaciones son diferentes. La idea de la balanza asume que los intereses de los editores y de los lectores difieren en importancia de forma sólo cuantitativa, en «cuánto al peso» que debemos darles y en qué situaciones se deben aplicar. El concepto de «la persona que guarda las apuestas» se suele usar para enmarcar la cuestión de este modo; supone que al tomar una decisión todos los intereses son igual de importantes. Este enfoque rechaza la distinción cualitativa entre los intereses de los lectores y de los editores que está en la base de la mediación gubernamental en el contrato de copyright.

Las consecuencias de esta alteración tienen un largo alcance, porque la fuerte protección que el público recibe con el contrato del copyright —a idea de que los privilegios del copyright sólo pueden justificarse en nombre de los lectores y nunca en el nombre de los editores— queda eliminada por esta interpretación de tipo «balanza». Dado que el interés de los editores es considerado como un fin en sí mismo, se puede justificar los privilegios del copyright; en otras palabras, el concepto de «balanza» dicta que los privilegios pueden justificarse en nombre de cualquiera que no sea el público.

En la práctica, la consecuencia del concepto de «balanza» es que invierte el peso de las justificaciones en lo que respecta a la legislación de copyright. El contrato de copyright coloca el peso en los editores para convencer a los lectores de que cedan ciertas libertades. El concepto de balanza prácticamente invierte este peso, porque en general no hay duda de que los editores se benefician de privilegios adicionales. De modo que, si no se prueba que el daño causado a los lectores es lo bastante grande como para «equilibrar» esos beneficios, se nos lleva a la conclusión de que los editores tienen derecho a casi cualquier privilegio que reclamen.

Dado que la idea de «equilibrar la balanza» entre editores y lectores niega a los lectores la primacía a la que tienen derecho, debemos rechazarla.

¿Qué se contraequilibra?

Cuando el gobierno compra algo para el público, actúa en su nombre; su responsabilidad es obtener el mejor trato posible —mejor para el público, no para la otra parte del acuerdo.

Por ejemplo, cuando firma contratos con las constructoras para hacer autopistas, el gobierno pretende gastar lo menor parte posible del erario público. Las agencias gubernamentales usan el sistema de concursos para forzar los precios a la baja.

En la práctica, el precio no puede ser igual a cero, porque los contratistas no pujarán tan bajo. Aunque no tengan derecho a consideraciones especiales, tienen los derechos ciudadanos corrientes en una sociedad libre, incluyendo el derecho a rechazar contratos no ventajosos; incluso la oferta más baja será suficiente para que algún contratista gane dinero. Luego ciertamente existe algún tipo de equilibrio. Pero no se trata de un equilibrio deliberado entre dos intereses que reclaman un trato especial. Es un equilibrio entre el bien público y las fuerzas del mercado. El gobierno trata de obtener para los contribuyentes automovilistas el mejor trato que puedan conseguir en el contexto de una sociedad libre y de un mercado libre.

En el contrato de copyright, el gobierno emplea nuestra libertad en lugar de nuestro dinero. La libertad es más valiosa que el dinero, así que la responsabilidad del gobierno en el empleo sabio y austero de nuestra libertad es incluso mayor que su responsabilidad en el uso de nuestro dinero. Los gobiernos jamás deben poner los intereses de los editores a la par que la libertad del público.

Mejor concesión que «equilibrio»

La idea de equilibrar los intereses de los lectores con los intereses de los editores es la forma más errónea de juzgar la política de copyright, pero ciertamente hay dos intereses que sopesar: dos intereses *de los lectores.* Los lectores están interesados en su propia libertad de consumir obras publicadas; y según las circunstancias, también estarán interesados en animar la publicación mediante algún tipo de incentivos.

La palabra «equilibrio», en las discusiones sobre copyright, ha quedado como una abreviatura de la idea de «equilibrar la balanza» entre lectores y editores. Por lo tanto, usar la palabra «equilibrio» a propósito de los dos intereses de los lectores sería confuso —necesitamos otro término.

En general, cuando un sujeto tiene dos objetivos que parcialmente entran en conflicto, y no puede realizar ambos por completo, llamamos a esto una concesión. Por lo tanto, mejor que hablar de «equilibrar la balanza» entre las partes, deberíamos hablar de «encontrar la concesión apropiada de libertad que considere tanto su pérdida necesaria como su conservación».

El segundo error: maximizar la producción

El segundo fallo de la política de copyright consiste en adoptar el objetivo de maximizar —no simplemente aumentar— la cantidad de obras publicadas. El concepto erróneo de «equilibrar la balanza» alzaba a los editores al nivel de los lectores; este segundo error los sitúa muy por encima de ellos.

Cuando adquirimos algo, por lo general no compramos toda la cantidad disponible ni tampoco el modelo más caro. En su lugar conservamos fondos para otras adquisicio-

nes, comprando sólo lo que necesitamos de un bien particular o escogiendo un modelo estándar antes que el de más alta calidad. El principio de los rendimientos decrecientes enseña que gastar todo nuestro dinero en un bien particular tiende a ser una ineficiente asignación de recursos; por lo general preferimos guardar algo de dinero para otro uso.

La ley de rendimientos decrecientes se ajusta al copyright tanto como a cualquier otra adquisición. Las primeras libertades que deberíamos ceder son aquellas que menos echamos de menos, al tiempo que damos el mayor respaldo a la publicación. Según cedemos libertades adicionales que se acercan más a lo que nos importa, encontramos que cada cesión supone un mayor sacrificio que la anterior, mientras que aporta un menor incremento a la actividad literaria. Antes de que el incremento sea igual a cero, bien podríamos decir que su creciente precio no merece la pena; entonces fijaríamos un contrato cuyo resultado general es incrementar la cantidad de lo publicado, pero no hasta su último extremo posible.

Aceptar el objetivo de maximizar la publicación supone rechazar de entrada todos estos contratos más ventajosos —este objetivo dispone que el público debe ceder casi toda su libertad de usar obras publicadas, a cambio de sólo unas pocas publicaciones más.

La retórica de la maximización

En la práctica, el objetivo de maximizar la producción sin que importe su coste para la libertad se sustenta en la extendida retórica que asegura que la copia pública es ilegal, ilegítima, injusta e intrínsecamente errónea. Por ejemplo, los editores llaman «piratas» a la gente que copia, término difamatorio pensado para equiparar el intercambio de información con tu vecina con el abordaje a un barco. (Este término difamatorio fue anteriormente usado por los autores para describir a los editores que encontraron formas legales de publicar ediciones no autorizadas; su uso moderno por parte de los editores es casi su reverso). Esta retórica rechaza directamente la base constitucional del copyright, pero se presenta a sí misma como representante de la incontestada tradición del sistema legal americano.

La retórica del «pirata» es aceptada frecuentemente en la misma medida en que ciega a los medios de comunicación, de tal modo que poca gente se da cuenta de su extremismo. Resulta efectiva porque si la copia por parte del público es fundamentalmente ilegítima, nunca podremos oponernos a los editores que exigen nuestra renuncia a la libertad de copiar. En otras palabras, cuando se reta al público a demostrar por qué los editores no deben recibir más poder, la razón más importante de todas —«queremos copiar»— es descalificada de entrada.

Esto no deja lugar para contestar el creciente poder del copyright sin entrar en cuestiones secundarias. Por lo tanto la oposición actual a un mayor poder del copyright alega exclusivamente cuestiones secundarias y nunca se atreve a alegar la libertad de distribuir copias como un valor público legítimo.

En concreto, el objetivo de la maximización capacita los editores para argumentar que «cierta práctica está reduciendo nuestras ventas —o pensamos que podría

reducirlas—, así que suponemos que reduce las publicaciones en una cantidad desconocida, y que por lo tanto debe ser prohibida». Se nos lleva a la espantosa conclusión de que el bien público se mide por las ventas de los editores: lo que es bueno para General Media es bueno para EE.UU.

El tercer error: maximizar el poder de los editores

Una vez que los editores han obtenido el consentimiento para el objetivo estratégico de maximizar la producción de publicaciones a cualquier coste, su próximo paso es probar que esto obliga a otorgarles los mayores poderes posibles —haciendo que el copyright cubra cualquier uso imaginable de una obra o aplicando cualquier otro instrumento legal como las licencias «de sobre cerrado» para conseguir un efecto equivalente. Este objetivo, que impone la abolición del «uso razonable» y el «derecho sobre la primera venta», está siendo objeto de presión en todos los niveles de gobierno imaginables, desde los estados de los EE.UU. hasta los organismos internacionales.

Esta medida es erróneo porque las reglas estrictas de copyright obstruyen la creación de nuevas obras útiles. Por ejemplo, Shakespeare tomó prestados los argumentos de algunas de sus obras teatrales de otras obras publicadas unas pocas décadas antes, de modo que de haber estado en funcionamiento la actual legislación de copyright, sus obras habrían sido ilegales.

Incluso si deseáramos el mayor grado posible de publicación, sin que importara el costo para el público, maximizar el poder de los editores es una forma errónea de conseguirlo. Como medio de promover el progreso, es autodestructivo.

Resultados de los tres errores

La tendencia actual en la legislación de copyright es proporcionar a los editores poderes más amplios por periodos de tiempo cada vez más largos. La base conceptual del copyright, en la medida en que resulta distorsionada por esta secuencia de errores, rara vez ofrece una base para decir no. Los legisladores defienden de boquilla la idea de que el copyright sirve al público, mientras que en realidad dan a los editores cualquier cosa que pidan.

Por ejemplo, esto es lo que dijo el senador Hatch al introducir la S. 483, una ley dictada en 1995 que incrementa la duración del copyright 20 años más:

> Creo que hemos llegado al punto de preguntarnos si la duración actual del copyright protege adecuadamente los intereses de los autores, y la cuestión correlativa de si el tiempo de protección sigue proporcionando suficientes incentivos para la creación de nuevas obras.

Esta ley extendió el copyright para obras publicadas y escritas desde 1920. Este cambio supuso una ganga para los editores, sin beneficio posible para el público, dado que ahora no hay modo de aumentar retroactivamente la cantidad de libros publicados

entonces. Sin embargo costó al público una libertad hoy muy significativa —la libertad de redistribuir libros de esa época.

La ley también extendió el copyright de las obras que aún no han sido escritas. Para trabajos de encargo, el copyright duraría 95 años en lugar de los 75 años que dura hoy. Teóricamente esto aumentaría los incentivos para escribir nuevos libros, pero cualquier editor que diga necesitar este incentivo extra debería apoyar esta pretensión con hojas de balance proyectadas hasta el año 2075.

No hace falta decir que el Congreso no cuestionó los argumentos de los editores: en 1998 fue promulgada la ley que extendía la duración del copyright. Fue conocida como la «ley Sonny Bono para la extensión de la duración del copyright», así llamada por uno de sus promotores, que había muerto ese año. Su viuda, que cubrió el resto de su trabajo, hizo esta declaración:

> En realidad, Sonny quería que el copyright durase para siempre. Mis abogados me han informado de que tal cambio violaría la Constitución. Os invito a todos vosotros a trabajar conmigo para reforzar nuestras leyes de copyright de todos los modos a nuestro alcance. Como sabéis, también está la propuesta de Jack Valenti para que dure para siempre menos un día. Quizá el comité pueda tratar este asunto el próximo Congreso.

El Tribunal Supremo ha admitido un pleito que busca derogar esta ley con el fundamento de que la retroactividad no sirve al objetivo constitucional de promover el progreso.

Otra ley, aprobada en 1996, convirtió en una fechoría hacer determinadas copias de cualquier obra publicada, incluso si vas a repartirlas entre amigos simplemente por amabilidad. Antes esto en absoluto era un crimen en los EE.UU.

Una ley todavía peor, la *Digital Millenium Copyright Act* (DMCA) fue diseñada para traer de vuelta la protección frente a las copias —que los usuarios de ordenadores detestan— convirtiendo en un crimen cualquier infracción de esta protección, o incluso publicar información sobre cómo quebrar esta protección. Esta ley debería de llamarse «ley para la dominación por parte de las empresas mass-mediáticas» porque ofrece efectivamente a los editores la oportunidad de escribir su propia ley de copyright. Dicta que pueden imponer cualquier tipo de restricciones en el uso de una obra, y estas restricciones tienen el rango de ley siempre que la obra contenga algún tipo de cifrado o gestor de licencias que haga efectivo su cumplimiento.

Uno de los argumentos ofrecidos por esta ley era que implementaría un reciente tratado para aumentar la extensión del copyright. Este tratado fue promulgado por la Organización Mundial de la Propiedad Intelectual (OMPI), organización dominada por los propietarios de los derechos de autor y de patentes, con la ayuda del gobierno de Clinton; dado que el tratado sólo aumenta la extensión del copyright, es dudoso que sirva al interés público en ningún país. En cualquier caso, la ley fue mucho más lejos de lo que el tratado demandaba.

Las bibliotecas fueron una fuente clave de oposición a esta ley, especialmente en los aspectos que coartan las formas de copia que se consideran de «uso razonable». ¿Cómo respondieron los editores? El antes diputado Pat Schroeder, ahora miembro del grupo

de presión a favor de la Asociación de Editores Americanos, dijo que los editores «no podrían vivir con lo que (las bibliotecas) están pidiendo». Dado que las bibliotecas sólo estaban pidiendo que se mantuviera parte del *statu quo*, uno podría contestar preguntando cómo habían sobrevivido los editores hasta entonces.

El congresista Barney Frank, durante una reunión conmigo y otros opositores a esta ley, demostró cuánto se ha menospreciado la visión que la Constitución de los EE.UU. tiene sobre el copyright. Dijo que se necesitan urgentemente nuevos poderes respaldados por nuevas penas, puesto que «la industria del cine está preocupada», así como la «industria de la música» y otras «industrias». Yo le pregunté, «¿pero todo esto es para el interés general?» Su respuesta vino a decir: «¿Por qué me hablas del interés general? ¡La gente creativa no tiene que abandonar sus derechos en favor del interés general!» La «industria» ha sido identificada con «la gente creativa» a la que contrata, el copyright ha sido tratado como su privilegio y la Constitución ha sido puesta patas arriba.

La DMCA fue publicada en 1998. Esta ley dice que el uso razonable sigue siendo legítimo, pero permite a los editores prohibir todo el software o el hardware con el que podrías ponerlo en práctica. De hecho, el uso razonable está prohibido.

Basándose en esta ley, la industria del cine ha impuesto la censura al software libre por leer y reproducir DVDs, e incluso a la información sobre cómo leerlos. En abril de 2001 el profesor Edward Felten, de la universidad de Princeton, fue intimidado bajo amenaza de una demanda judicial por parte de la Asociación de la Industria Americana de Grabación, con el objetivo de que retirara un artículo científico en el que exponía lo que había aprendido acerca de un sistema piloto de encriptación que restringía el acceso a las grabaciones musicales.

También estamos empezando a ver libros electrónicos que retiran a los lectores muchas de sus libertades tradicionales —por ejemplo, la libertad de prestar un libro a un amigo, de venderlo a una tienda de libros usados, de tomarlo prestado de una biblioteca, de comprarlo sin darle tu nombre a una base de datos empresarial, incluso la libertad de leerlo dos veces. Los libros electrónicos codificados restringen por lo general este tipo de actividades —sólo puedes leerlos con un software especial secreto diseñado para limitar tus libertades.

Nunca compraré uno de esos libros electrónicos, cifrados y restrictivos, y espero que tú también los rechaces. ¡Si un libro electrónico no te da las mismas libertades que un libro tradicional de papel, no lo aceptes!

Cualquiera que lance por su cuenta software que pueda leer libros electrónicos restrictivos se arriesga a ser acusado. Un programador ruso, Dmitry Sklyarov, fue arrestado en 2001 mientras visitaba EE.UU. para dar en una conferencia, porque había escrito un programa de esas características en Rusia, donde hacerlo era legal. Ahora Rusia también está preparando una ley para prohibirlo y la Unión Europea ha adoptado una similar recientemente.

Los libros electrónicos para el mercado de masas han resultado hasta ahora un fracaso comercial, pero no porque los lectores elijan defender su libertad; no eran atractivos por otras razones, como el hecho de que los monitores de los ordenadores no son soportes cómodos para la lectura. No podemos confiar en esta feliz casualidad para

protegernos a largo plazo; el próximo intento para promocionar los libros electrónicos usará «papel electrónico» —objetos parecidos a libros dentro de los cuales se puede descargar un libro electrónico codificado y restringido. Si este soporte parecido al papel demuestra ser más atractivo que los actuales monitores, tendremos que defender nuestra libertad para conservarla. Mientras tanto, los libros electrónicos están penetrando nichos de mercado: la NYU y otras escuelas de odontología obligan a sus estudiantes a comprar sus libros de texto con formato de libros electrónicos restrictivos.

Las empresas mediáticas no están satisfechas todavía. En 2001, el senador Hollings —financiado por la Disney— propuso una ley llamada «ley para los estándares de los sistemas de seguridad y de las certificaciones», [3] que obligaría a que todos los ordenadores —y otros dispositivos de grabación y reproducción digital— tuvieran sistemas restrictivos de copia por mandato del gobierno. Ese es su objetivo final, pero el primer punto de su agenda es prohibir cualquier equipo que pueda sintonizar HDTV digital a no ser que esté diseñado para que al público le sea imposible «entrometerse» —por ejemplo, modificarlo para sus propios fines. Dado que el software libre es software que los usuarios pueden modificar, aquí nos enfrentamos por primera vez con una propuesta de ley que prohíbe explícitamente el software libre para un trabajo determinado. Seguramente, le seguirá la prohibición de otros trabajos. Si el FCC adopta esta regla, el software libre hoy existente, como GNU Radio, sería censurado.

La acción política es necesaria para bloquear estas leyes y reglamentos. [4]

Encontrar el contrato adecuado

¿Cuál es la manera adecuada de decidir la política de copyright? Si el copyright es un contrato hecho en nombre del público, debería servir ante todo al interés público. El deber del gobierno al vender la libertad del público es vender sólo lo que debe y venderlo tan caro como sea posible. Como mínimo, deberíamos recortar, en la medida de lo posible, el alcance del copyright mientras podamos un nivel comparable de publicación.

Dado que no podemos encontrar este precio mínimo en términos de libertad a través de un concurso público como se hace con los proyectos de construcción, ¿qué podemos hacer?

Un método posible es reducir los privilegios del copyright por etapas y observar los resultados. Observando si hay disminuciones apreciables en el volumen de publicación, aprenderemos que extensión del copyright es realmente necesaria para llevar a cabo los propósitos del público. Esto se debe valorar por medio de la observación práctica, no por lo que los editores digan que ocurrirá, ya que tienen todos los motivos para hacer predicciones exageradas de perdidas si sus poderes se ven reducidos de algún modo.

[3] Luego rebautizada con el nombre impronunciable de LPCBATD, para la cual «consume, pero no intentes programar nada» es un buen recordatorio; pero las siglas significan realmente Ley de Promoción del Consumo de Banda Ancha y Televisión Digital.

[4] Si quieres ayudar, te recomiendo los sitios web `digitalspeech.org` y `www.eff.org`.

La política de copyright tiene varias dimensiones independientes que pueden ajustarse de forma separada. Después de encontrar el mínimo necesario para una vertiente de esta política, todavía sería posible reducir otras dimensiones del copyright a la vez que se mantiene el nivel de publicación deseado.

Una dimensión importante del copyright es su duración, que ahora se encuentra en torno a un siglo de media. Reducir el monopolio sobre la copia a diez años, desde la fecha en que una obra es publicada, sería un buen primer paso. Otro aspecto del copyright, que cubre la realización de obras derivadas, podría extenderse por más tiempo.

¿Por qué contar desde la fecha de publicación? Por que el copyright de las obras no publicadas no limita directamente la libertad los lectores; que tengamos libertad para copiar una obra es una cuestión inútil cuando aun no tenemos una copia. Así que dar a los autores más tiempo para publicar un trabajo no hace ningún daño. Los autores —que por lo general sí poseen el *copyright* antes de publicar— rara vez elegirán retrasar la publicación sólo para alejar el fin del plazo del copyright.

¿Por qué diez años? Porque esta es una propuesta segura; podemos confiar en el terreno práctico que esta reducción tendrá, hoy en día, poco impacto en la viabilidad general de la edición. En muchos medios y géneros, las obras de éxito son muy rentables unos pocos años, y normalmente incluso las obras de éxito ya no se editan pasados los diez años. Incluso para las obras de referencia, cuya vida útil puede ser de muchas décadas, el copyright de diez años debería de bastar: las ediciones actualizadas se lanzan con regularidad y muchos lectores comprarán la última versión con copyright antes que copiar una versión de dominio público con diez años de antigüedad.

Diez años todavía puede ser más tiempo del necesario; una vez que las cosas se estabilicen, podremos probar mayores reducciones para ajustar el sistema. En una mesa redonda sobre copyright durante una convención literaria, en la que yo propuse el plazo de diez años, un famoso escritor de fantasía que se sentaba junto a mí protestó vehementemente, diciendo que cualquier cosa que sobrepasara los cinco años era intolerable.

Pero no tenemos por qué aplicar el mismo lapso de tiempo para todas las obras. Mantener la uniformidad extrema de las políticas de copyright no es crucial para el interés público y la legislación de copyright ya incluye muchas excepciones para medios y usos específicos. Sería estúpido pagar por cada proyecto de autopista al precio de los proyectos más difíciles y en las regiones más caras del país; es igualmente estúpido «pagar» por todo tipo de arte el precio más alto, en términos de libertad, que consideramos necesario para un caso determinado.

Así, quizás las novelas, los diccionarios, los programas informáticos, las canciones, las sinfonías y las películas deberían tener un copyright con distintas duraciones, de modo que podamos reducir la duración en cada tipo de obra a lo que sea necesario para ese tipo de obras se publiquen. Quizá las películas con duración mayor de una hora podrían tener un copyright de 20 años, debido a los costes de producción. En mi propio campo, la programación informática, tres años deberían bastar, dado que los ciclos de un producto son incluso más cortos.

Otra dimensión de la política de copyright es la magnitud del uso razonable: algunas formas legalmente permitidas de reproducción de un trabajo, total o parcialmente, aún cuando éste está protegido por el copyright. El primer paso natural para reducir este aspecto del copyright es permitir la copia privada sin ánimo de lucro, ocasional y en pequeña cantidad, para su distribución entre individuos. Esto eliminaría la intrusión de la policía del copyright en la vida privada de la gente, pero probablemente tendría poco efecto en las ventas de las obras publicadas. (Podría ser necesario tomar otras medidas legales para asegurar que las licencias de uso de «sobre cerrado» no sean usadas para sustituir al copyright y restringir este tipo de copia.) La experiencia de Napster muestra que también deberíamos permitir al público general la redistribución textual y no comercial —cuando tanta gente entre el público quiere copiar y compartir, y lo encuentra tan útil, sólo conseguirán detenerlo medidas draconianas; el público tiene derecho a obtener lo que quiere.

Para las novelas, y en general para las obras que se utilizan como entretenimiento, la redistribución textual no comercial podría ser una libertad suficiente para los lectores. Los programas informáticos, al ser usados para fines funcionales —para trabajar—, exigen libertades adicionales, incluyendo la libertad de publicar una versión mejorada. (Consulta la definición de «software libre» en este libro para una explicación de las libertades que los usuarios de software deberían de tener.) Sin embargo, en relación a estas libertades un compromiso aceptable podría ser que estuvieran disponibles universalmente únicamente después de una retraso de dos o tres años con respecto a la publicación del programa.

Cambios como estos podrían adaptar el copyright al deseo del público de usar la tecnología digital para copiar. Sin duda los editores encontrarán estas propuestas «desproporcionadas»; podrían amenazar con recoger «sus fichas y largarse del juego», pero en realidad no lo harán, porque el juego seguirá siendo rentable y será el único juego posible.

Al igual que consideramos la reducción de la extensión del copyright, debemos asegurarnos de que simplemente las empresas mediáticas no lo reemplazarán con acuerdos de licencia para el usuario final. Sería necesario prohibir el uso de contratos que aplican a la copia restricciones que van más allá que las reguladas por el copyright. Dichas limitaciones, que pueden ser prescritas por los contratos no negociados del mercado de masas, son una parte estándar del sistema legal de los EE.UU.

Una nota personal

Soy programador de software, no un experto en derecho. He llegado a interesarme por el copyright porque no hay forma de evitarlo en el mundo de las redes informáticas.[5] Como usuario de ordenadores y de redes desde hace treinta años, valoro las libertades que hemos perdido y las que podríamos perder. Como autor, puedo rechazar la mistificación romántica del autor como creador semidivino, frecuentemente esgrimida

[5] Siendo Internet la más grande de las redes informáticas mundiales

por los editores para justificar el incremento en el alcance del copyright de los autores, que los autores luego cederán a los editores.

La mayor parte de este artículo se refiere a hechos y razonamientos que puedes comprobar, propuestas sobre las que te puedes formar tus propias opiniones. Pero te pido que aceptes algo que sólo se basa en mi palabra: que los autores como yo no tenemos derecho a ningún poder especial sobre ti. Si deseas recompensarme más por el software o los libros que he escrito, aceptaré agradecido un cheque, pero por favor no entregues tu libertad en mi nombre.

Capítulo 13
La ciencia debe desechar el copyright[1]

Debería ser un axioma que la literatura científica existe para divulgar el conocimiento científico, y que las revistas científicas existen para facilitar este proceso. Por consiguiente, las reglas de uso de la literatura científica deberían diseñarse para ayudar a conseguir este objetivo.

Las reglas que tenemos ahora, conocidas como copyright, fueron establecidas en la era de la imprenta, un método intrínsecamente centralizado para la producción masiva de copias. En el contexto de la imprenta, el copyright sobre los artículos de publicaciones sólo restringía a los editores, obligándoles a obtener un permiso para publicar un artículo, y a los posibles plagiarios. Esto ayudó a que las revistas activaran y divulgaran el conocimiento sin interferir en el provechoso trabajo de los científicos o estudiantes, ya sea como escritores o como lectores de artículos. Estas reglas se adecuaban bien a dicho sistema.

La tecnología moderna para las publicaciones científicas es, sin embargo, Internet. ¿Qué reglas asegurarían mejor la divulgación de los artículos científicos y del conocimiento en la Red? Los artículos deberían de distribuirse en formatos no propietarios, de acceso abierto para todos. Y todos deberían de tener el derecho de reproducir los artículos, esto es, de reeditarlos íntegramente con su adecuada atribución.

Estas reglas deberían aplicarse tanto a los artículos pasados como a los futuros, cuando se distribuyen en formato digital. Pero no hay ninguna necesidad crucial de cambiar el sistema de copyright actual aplicado a la edición impresa de revistas, porque el problema no afecta a ese dominio.

Por desgracia, parece que no todo el mundo está de acuerdo con los axiomas que encabezan este artículo. Muchos editores de revistas parecen creer que el propósito de la literatura científica es permitirles editar revistas para cobrar suscripciones de científicos y estudiantes. Esta forma de pensar se conoce como «confundir los medios con los fines».

Su proceder ha consistido en restringir el acceso a la lectura de literatura científica, incluso a aquellos que pueden pagar y que pagarán por ello. Usan la legislación de copyright, todavía vigente a pesar de su inadecuación a las redes informáticas, como una excusa para detener a los científicos en la selección de nuevas reglas.

[1] Publicado originalmente en la página web de Nature.com, en su sección «Debates».

En nombre de la cooperación científica y del futuro de la humanidad, debemos rechazar tal enfoque desde su raíz —no sólo los sistemas restrictivos que se han establecido, sino las prioridades equivocadas que los inspiraron.

Los editores de revistas a veces argumentan que el acceso *on line* requiere servidores caros de alta capacidad y que deben cobrar tarifas de acceso para pagar estos servidores. Este «problema» es una consecuencia de su propia «solución». Concede a todo el mundo la libertad de autoeditar, y las bibliotecas en todo el mundo montarán páginas de libre publicación para responder a la demanda. Esta solución descentralizada reducirá las necesidades de ancho de banda de la red y proveerá un acceso más rápido, a la vez que se protege la documentación académica contra pérdidas accidentales.

Los editores también sostienen que pagar a los encargados de la página obliga a cobrar por el acceso. Aceptemos la suposición de que los encargados deben ser pagados; para este viaje no hacen falta alforjas. El coste de la edición de un revista normal está entre el uno y el tres por ciento del coste de financiar la investigación para producirla. Un porcentaje tan pequeño difícilmente puede justificar que se obstaculice el uso de los resultados.

En su lugar, el coste de la edición puede cubrirse, por ejemplo, cobrando a los autores por publicar en la página, y estos pueden traspasar estos pagos a los patrocinadores de su investigación. A los patrocinadores no les debería de importar, dado que ya pagan por la publicación de una forma más molesta, a través de las tarifas astronómicas que abonan para que la biblioteca universitaria se suscriba a la revista. Mediante el cambio de modelo económico para que los patrocinadores de la investigación cubran los costes de la edición, podemos eliminar la necesidad aparente de restringir el acceso. El autor fortuito que no pertenece a ninguna institución o empresa, y que no tiene patrocinador, podría estar exento de estos pagos, con los costos derivados a los autores patrocinados.

Otra justificación para las tarifas de acceso a las publicaciones de Internet es que pueden financiar la reconversión de los archivos impresos de un revista a un formato *on line*. Este trabajo tiene que hacerse, pero deberíamos buscar formas alternativas de financiarlo que no supongan obstruir el acceso a los resultados. El trabajo en sí mismo no será más difícil ni costará más. Digitalizar los archivos y malgastar los resultados por restringir el acceso a ellos es algo autodestructivo.

La Constitución de los EE.UU. dice que el copyright existe «para promover el progreso de la ciencia». Cuando el copyright impide el progreso de la ciencia, la ciencia debe desechar el copyright.

Capítulo 14
¿Qué es el copyleft?[1]

El copyleft es un método para convertir un programa en software libre y exigir que todas las versiones del mismo, modificadas o ampliadas, también lo sean.

La forma más sencilla de hacer que un programa sea libre es ponerlo en el dominio público, sin derechos reservados. Esto permite a la gente compartir el programa y sus mejoras, si así lo desean. Pero asimismo permite, a quienes no crean en la cooperación, convertir el programa en software propietario. Pueden hacer cambios, muchos o pocos, y distribuir su resultado como un producto propietario. Las personas que reciben el programa con esas modificaciones no gozan de la libertad que les dio el autor original; el intermediario les ha despojado de ella.

En el proyecto GNU, nuestro objetivo es proporcionarle a todos los usuarios la libertad para redistribuir y modificar el software GNU. Si los intermediarios pudieran eliminar esa libertad, nosotros veríamos aumentar nuestro número de usuarios, pero esos usuarios no dispondrían de libertad. Así que, en vez de poner software GNU en el dominio público, lo protegemos con copyleft. De acuerdo con el copyleft, cualquiera que distribuya software, con o sin modificaciones, debe traspasar con él la libertad para copiarlo y modificarlo. El copyleft garantiza que cada usuario goce de esta libertad.

El copyleft también incentiva a otros programadores a introducir mejoras en el software libre. Programas importantes como el compilador GNU para C++ existen gracias a esto.

El copyleft también ayuda a programador o a la programadora que deseen contribuir a mejorar el software libre al darles autorización para ello. Estos programadores o estas programadoras a menudo trabajan para empresas o universidades que harían casi cualquier cosa para obtener más dinero. Un programador o una programadora puede querer aportar sus cambios a la comunidad, pero su empresa preferirá convertir sus modificaciones en un producto de software propietario.

Cuando le explicamos a la empresa que es ilegal distribuir la versión mejorada a menos que sea en forma de software libre, normalmente ésta optará por distribuirla como software libre antes que desecharla.

Para aplicar el copyleft a un programa, primero reservamos los derechos; luego añadimos los términos de distribución, un instrumento legal que otorga a todo el mundo el derecho a utilizar, modificar y redistribuir el código del programa o cualquier progra-

[1] Escrito originalmente en 1996.

ma derivado del mismo, siempre que no se alteren los términos de distribución. De esta forma, el código y las libertades se convierten en elementos legalmente inseparables.

Los desarrolladores de software propietario usan el copyright para restar libertad a los usuarios; nosotros recurrimos a los derechos reservados para garantizársela. Por eso invertimos el nombre, convirtiendo los derechos reservados —*copyright*— en copyleft.

El copyleft es un concepto general. Hay muchas maneras de interpretarlo. En el proyecto GNU, los términos de distribución específicos que utilizamos están contenidos en la General Public License GNU (GNU GPL). La General Public License GNU es llamada muchas veces GNU-GPL para abreviar. Existe una página de consulta[2] sobre la GNU GPL. También puedes leer por qué la FSF obtiene la cesión de los derechos de *copyright* de aquellos que quieren contribuir con ella[3].

Una forma alternativa, la Lesser General Public License o Licencia Pública General para Bibliotecas GNU (GNU LGPL), se aplica a algunas —que no a todas— de las bibliotecas GNU. Esta licencia solía llamarse Library GPL, pero la rebautizamos porque el nombre anterior invitaba al uso indiscriminado de esta licencia. Para más detalles, de por qué este cambio era necesario, véase «Por qué no deberías utilizar la Library GPL en tu próxima biblioteca».

La Lesser General Public License GNU sigue disponible en HTML, aunque ha sido reemplazada oficialmente por la licencia arriba indicada.

La licencia apropiada se incluye en muchos manuales y en cada distribución de código fuente GNU.

La GNU Free Documentation License FDL es una forma de copyleft diseñada para manuales, libros de texto u otros documentos, que asegura a cualquiera la libertad de copia y de distribución, con o sin modificaciones, ya sea en de forma comercial o no comercial.

La GPL GNU está diseñada para que puedas aplicarla fácilmente en tu propio programa siempre y cuando poseas derechos sobre él. No tienes que modificar la GPL GNU para hacerlo, basta con añadir una nota en tu programa que haga referencia a ella.

Si desearas aplicar el copyleft a tu programa con GPL/GNU, lee las instrucciones al final del texto de la GPL[4]. Por favor, considera que debes utilizar el texto completo de la GPL. Es un conjunto íntegro y las copias parciales no están permitidas —de igual modo que con la LGPL.

Emplear los mismos términos de distribución para muchos programas diferentes facilita la copia del código entre varios programas. Ya que todos comparten idénticos términos de distribución, no es necesario preocuparse por si los términos son compatibles o no. La LGPL permite además alterar los términos de distribución de la GPL ordinaria, de modo que pueda copiarse el código dentro de otro programa cubierto por la GPL.

[2] `http://www.gnu.org/licenses/gpl-faq.html`

[3] `http://www.gnu.org/copyleft/why-assign.html`

[4] `http://www.gnu.org/copyleft/gpl-howto.html`

Sí deseas poner un copyleft en tu manual con la GNU-LDL, por favor sigue las instrucciones al final del texto de esa licencia, y las instrucciones de la página GFDL[5]. Como en el caso de la GNU-GPL, debes usar la licencia completa, no están permitidas las copias parciales.

[5] http://www.gnu.org/copyleft/fdl-howto.html

Capítulo 15

Copyleft: idealismo pragmático

[1]

Toda decisión que una persona toma entronca con sus valores y objetivos. La gente pude tener objetivos y valores muy distintos: la fama, el dinero, el amor, la supervivencia, la diversión y la libertad sólo son algunos de los objetivos que una buena persona puede tener. Cuando el objetivo también es ayudar a los otros y a uno mismo, lo llamamos idealismo.

Mi trabajo con el software libre está motivado por un objetivo idealista: difundir la libertad y la cooperación. Quiero promover la difusión del software libre, sustituyendo al software propietario que prohíbe la cooperación, para de este modo mejorar nuestra sociedad.

Esa es la razón principal de que la Licencia Pública General de GNU esté escrita tal y como lo está —como copyleft. Todo código añadido a programas protegidos por la GPL debe ser software libre, incluso si se coloca en un archivo separado. Hago que mi código esté disponible como software libre y no como software propietario, para animar a otra gente que escribe software a que también lo haga. Supongo que como los desarrolladores de software propietario utilizan el copyright para que no podamos compartir, los que cooperamos podemos usar el copyright para dar a otros que cooperan una ventaja propia: la de usar nuestro código.

No todos los que usan GNU GPL tienen este objetivo. Hace muchos años, a un amigo mío le pidieron que redistribuyera un programa copyleft en condiciones que no eran copyleft, y respondió más o menos así: «A veces trabajo con software libre y otras con software propietario pero cuando trabajo con software propietario, espero que me paguen».

Quería compartir su trabajo con una comunidad que compartiese software, pero no veía ninguna razón para hacer una donación a un negocio que fabrica productos fuera del alcance de nuestra comunidad. Su objetivo era diferente del mío, pero decidió que GNU GPL también era útil para su objetivo.

Si quieres lograr algo en este mundo, el idealismo no es suficiente —necesitas escoger un método que funcione para conseguir tu objetivo—. En otras palabras, necesitas ser «pragmático». ¿Es pragmática la GPL? Echemos un vistazo a sus resultados.

[1] Este artículo fue escrito originalmente en 1998.

Consideremos GNU C++. ¿Por qué tenemos un compilador libre C++? Sólo porque la GNU GPL dicta que tiene que ser libre. GNU C++ fue desarrollado por un consorcio industrial, MCC, a partir del compilador GNU C. Normalmente la MCC hace su trabajo todo lo propietario que puede. Pero hicieron el front end[2] C++ con software libre porque la GNU GPL dictaba que era el único modo en que podían publicarlo. El *fron end* C++ incluía muchos archivos nuevos, pero dado que supuestamente tenían que estar relacionados con GCC,[3] la GPL encajaba en ellos. El beneficio para nuestra comunidad fue evidente.

Consideremos GNU Objective C. Inicialmente NeXT[4] quiso hacer propietario este *front end*; propusieron que se lanzara como un archivo «.o», y dejar que los usuarios lo enlazaran con el resto de GCC, pensando que esta sería una aproximación a las exigencias de la GPL, pero nuestro abogado dijo que esto no salvaba los requerimientos legales, que no estaba permitido. Por eso hicieron libre el *front end* Objective C.

Estos casos tuvieron lugar hace años, pero la GNU GPL sigue proporcionándonos más software libre.

Muchas bibliotecas de GNU están protegidas por la licencia pública general para bibliotecas, pero no todas. Una biblioteca de GNU protegida por la licencia GPL normal es Readline, que implementa la edición de líneas de comandos. Una vez encontré un programa no libre que estaba diseñado para usar Readline, y le dije al desarrollador que esto no estaba permitido. Podría haber eliminado del programa la edición de líneas de comandos, pero lo que hizo en realidad fue redistribuirlo bajo GPL. Ahora es software libre.

Los programadores que escriben mejoras para GCC —o Emacs, o Bash, o Linux, o cualquier programa protegido por la GPL— frecuentemente son empleados de empresas o universidades. Cuando el programador quiere devolver sus mejoras a la comunidad y muestra su código en la siguiente publicación, el jefe le dirá:

> Espera un momento. ¡Tu código nos pertenece! No queremos compartirlo; hemos decidido convertir tu versión mejorada en un producto de software propietario.

Aquí la GNU GPL viene a nuestro rescate. El programador enseña al jefe que este producto de software propietario infringiría el copyright y el jefe se da cuenta de que sólo tiene dos opciones: publicar el nuevo código como software libre o no publicarlo en absoluto. Casi siempre permite que el programador haga lo que pretendía desde el principio y el código irá incluido en el siguiente lanzamiento.

---

[2]El *front end* es la parte del compilador que analiza el código fuente, comprueba su validez, genera el árbol de derivación y rellena los valores de la tabla de símbolos. Esta parte suele ser independiente de la plataforma o sistema para el cual se vaya a compilar. EL término se emplea normalmente en inglés, aunque alguna vez se traduce por «frontal», se trata de un falso amigo, sería más preciso algo como «antecesor», «precursor» o incluso «pre-procesador». [*N. del E.*]

[3]El compilador GNU del lenguaje C. [*N. del E.*]

[4]Un sistema operativo creado por Steve Jobs, finalmente comprado por Apple.

La GNU GPL no es una hermanita de la caridad. Impide algunas cosas que la gente a veces quiere hacer. Hay usuarios que dicen que esto es un mal asunto —que la GPL «excluye» a algunos creadores de software propietario que «es necesario introducir en la comunidad del software libre».

Pero nosotros no los excluimos de nuestra comunidad; ellos eligen no entrar. Su decisión de hacer software propietario es una decisión de mantenerse fuera de nuestra comunidad. Estar en nuestra comunidad significa unirse a nosotros por medio de la cooperación; no podemos «introducirlos en nuestra comunidad» si ellos no quieren unirse.

Lo que *podemos* hacer es ofrecerles un aliciente para unirse. La GNU GPL está diseñada para hacer de nuestro software disponible un aliciente: «si haces libre tu software, puedes usar este código». Por supuesto, no te ganas a todos, pero a veces te ganas a algunos.

El desarrollo del software propietario no ayuda a nuestra comunidad, pero a menudo sus creadores quieren donaciones de nuestra parte. Los usuarios de software libre pueden proporcionar autoestima —reconocimiento y gratitud— a los creadores de software libre, pero también puede ser muy tentador cuando una empresa te dice: «¡Tú deja simplemente que metamos tu paquete en nuestro programa propietario y tu programa lo usarán muchos miles de personas!» La tentación pude ser poderosa, pero a largo plazo todos estaremos en mejor situación si nos resistimos a ella. La tentación y la presión resultan difíciles de reconocer si llegan de forma indirecta, a través de organizaciones de software libre que han adoptado una política de provisión de software propietario. El X Consortium —y su sucesor, el Open Group— ofrece un ejemplo: financiados por compañías que producían software propietario, se han afanado, durante una década, en persuadir a los programadores de que no usen copyleft. Ahora que el Open Group ha hecho de X11R6.4 software no libre, los que resistimos esa presión estamos encantados de haber aguantado.[5]

En términos prácticos, pensar en objetivos a largo plazo reforzará tu voluntad de resistir a esta presión. Si piensas en la libertad y en la comunidad que puedes construir permaneciendo firme, encontrarás la fuerza para hacerlo. «Resiste por algo o caerás por nada».

Y si los cínicos ridiculizan la libertad, si ridiculizan a la comunidad... si los «implacables realistas» dicen que las ganancias son el único ideal... simplemente ignóralos y sigue usando el copyleft.

[5]En septiembre de 1998, varios meses después de que X11R6.4 fuera publicado en condiciones de distribución no libre, el Open Group trastocó su decisión y lo relanzó bajo la misma licencia de software libre no copyleft que se usó para X11R6.3. Gracias, Open Group, pero este trastocamiento posterior no invalida las conclusiones que establecimos a partir del hecho de que fuera posible añadir restricciones.

Capítulo 16

El peligro de las patentes de software[1]

Posiblemente estéis familiarizados con mi trabajo sobre el software libre. Esta charla no trata sobre ese trabajo. Esta charla trata sobre una forma de abuso legal para hacer del desarrollo informático una actividad peligrosa. Esto es, más o menos, lo que pasa cuando una ley de patentes se aplica al campo del software.

No trata del hecho de patentar el software. Esta es una forma muy mala, una forma muy engañosa de describir la cuestión, porque el problema no está en patentar programas individuales. Si así fuera, no habría ninguna diferencia, sería algo básicamente inocuo. En vez de eso, esta charla trata sobre el hecho de patentar ideas. Toda patente protege alguna idea. Las patentes de software son patentes que protegen ideas que tienen que ver con el software, ideas que podrían usarse para desarrollar software. Eso es lo que las convierte en peligrosos obstáculos para cualquier desarrollo de software.

Quizá hayáis oído a la gente utilizar un término engañoso, «propiedad intelectual». Este término, como veréis, está sesgado: asume que, digas lo que digas, la forma de considerar el software está en relación a algún tipo de propiedad, cuando esta forma en realidad es una entre muchas otras alternativas. Este término, «propiedad intelectual», prejuzga la cuestión más básica en cualquiera de las áreas que consideréis. No contribuye a despejar y abrir la mente.

Existe un problema adicional con el término, que no tiene nada que ver con el desarrollo de la opinión que tenga cada uno; y es que impide la comprensión de los hechos. El término «propiedad intelectual» vale para todo, mezcla aspectos completamente dispares de la ley, como puedan ser el copyright y las patentes que son completamente distintos. Cada detalle es singular. También mezcla la cuestión de las marcas, que todavía genera más diferencia y otras cosas que se encuentran con menos frecuencia. Ninguna de ellas tiene nada en común con cualquiera de las otras. Históricamente sus orígenes están completamente separados; las leyes se diseñaron de forma independiente; cubrían diferentes actividades y aspectos de la vida. Las medidas políticas que crearon están completamente desconectadas, de modo que si intentáis pensar en ellas confundiéndolas, tendréis la garantía de llegar a conclusiones disparatadas. Literalmente no podéis tener una opinión sensata ni inteligente sobre la «propiedad intelectual». Por lo tanto, si queréis pensar con claridad, no mezcléis estas cuestiones. Pensad sobre el copy-

[1]Esta charla fue pronunciada en la Universidad de Cambridge, Londres, el 25 de marzo de 2002.

right y luego pensad sobre las patentes. Aprended acerca de la legislación de copyright y de forma separada aprended acerca de la legislación de patentes.

Por citar algunas de las diferencias más grandes entre el copyright y las patentes:

- El copyright regula las condiciones de expresión de una obra, no protege ninguna idea. Las patentes sólo protegen las ideas y el uso de las ideas.
- El copyright se aplica automáticamente. Las patentes son publicadas por una oficina de patentes como respuesta a una solicitud.
- Las patentes cuestan mucho dinero. Cuestan más por lo que se paga a los abogados para que realicen la solicitud, que por lo que realmente cuesta su aplicación. Normalmente la solicitud tarda algunos años en ser estudiada, aún cuando las oficinas de patentes realizan un trabajo de estudio extremadamente precario.
- El copyright dura durante un tiempo extremadamente largo. En algunos casos puede durar hasta 150 años. Las patentes duran 20 años, lo cual es suficiente como para que sobrevivas a su caducidad, pero todavía es bastante tiempo con respecto a la escala de un campo como el software. Pensemos en relación a hace 20 años, cuando el PC era algo novedoso. Imaginad que estuviéramos limitados a desarrollar software utilizando únicamente las ideas conocidas en 1982.
- El copyright sólo protege la copia. Si escribes una novela que resulta ser igual palabra por palabra a «Lo que el viento se llevó» y puedes probar que nunca has visto «Lo que el viento se llevó», bastaría como defensa contra cualquier acusación de haber infringido el copyright.
- La patente es un monopolio absoluto sobre el uso de una idea. Incluso si pudieras probar que la idea es tuya, sería completamente irrelevante si la idea ha sido patentada por otro.

Espero que os olvidéis del copyright en lo que queda de exposición, porque esta exposición trata sobre patentes y nunca se deben mezclar las patentes y el copyright, si se quiere comprender claramente estos dos asuntos.

Imaginad que pasaría si al estudiar química práctica —o cocina— confundierais el agua con el etanol.

Cuando escuchas a la gente describir el sistema de patentes, normalmente lo hacen desde el punto de vista de alguien que espera conseguir una patente —cómo sería para ti conseguir una patente, como sería andar por la calle con una patente en tu bolsillo, para poder sacarla cada dos por tres, mostrársela a alguien y decir «¡dame tu pasta!».

Hay un motivo para este prejuicio: la mayoría de la gente que habla del sistema de patentes ha apostado por él, y por lo tanto quiere seduciros. Hay otro motivo: el sistema de patentes se parece mucho a la lotería, sólo una fracción muy pequeña de las patentes reporta realmente algún beneficio a aquellos que las poseen. De hecho, The Economist comparó una vez este sistema con una «lotería que consume tiempo». Si has visto anuncios de lotería, siempre te incitan a pensar que vas a ganar. No te incitan a

pensar que vas a perder, aunque perder es mucho más probable. Con la propaganda del sistema de patentes pasa lo mismo: siempre te incitan a pensar que vas a ganar.

Para compensar este prejuicio, voy a describir el sistema de patentes desde el punto de vista de sus víctimas —esto es, desde el punto de vista de alguien que quiere desarrollar software pero está obligado a un forcejeo con un sistema de patentes informáticas que puede llevarle a ser demandado.

Por lo tanto, ¿qué es lo primero que puedes hacer después de haber tenido una idea sobre el tipo de programa que quieres escribir?

Para tratar con el sistema de patentes, lo primero que podrías intentar es descubrir qué patentes pueden cubrir el programa que quieres escribir. Esto es, sin embargo, imposible.

La razón se encuentra en que algunas de las solicitudes de patentes en trámite son secretas. Pasado cierto tiempo, 18 meses, podrán publicarse. Sin embargo, 18 meses son tiempo de suficiente para que escribas el programa, e incluso para lanzarlo sin saber que existe una patente y que vas a ser demandado.

No es un asunto simplemente académico. En 1984 se escribió el programa Compress, un programa para la compresión de datos. En esa fecha, no había una patente para el algoritmo LZW de compresión que usaba. Más tarde, en 1985, los EE.UU. publicaron una patente sobre este algoritmo y durante los años siguientes los que distribuían el programa Compress empezaron a recibir amenazas.

No había forma de que el autor de Compress se hubiera dado cuenta de que podía ser demandado. Todo lo que hizo fue usar una idea que encontró en una revista, como siempre habían hecho los programadores. No se había dado cuenta de que ya no se podían usar de forma segura las ideas que encontrabas en una revista.

Olvidemos ese problema. Las patentes en curso son publicadas por la oficina de patentes, de modo que puedes encontrar la lista completa y ver qué dictan exactamente.

Por supuesto, en realidad no podrías leer toda la lista, ya que hay demasiadas patentes. En EE.UU. hay cientos de miles de patentes de software. No hay forma de que puedas seguirles la pista de todo lo que contienen. Tendrías que intentar la búsqueda de las más importantes.

Algunos dicen que eso debería ser fácil en la moderna era del ordenador. Podrías buscar a partir de palabras clave, pero eso sólo funciona hasta cierto punto. Encontrarás algunas patentes en una determinada. Pero probablemente no encontrarás todas.

Por ejemplo, existe una patente de software —que quizá ya haya expirado— sobre el cálculo en orden natural para hojas de cálculo. Básicamente, esto quiere decir que cuando produces una celda dependiente de otra celda, todo se vuelve a calcular en función de aquello de lo que depende, de modo que después de una operación de cálculo todo queda actualizado. Las primeras hojas de cálculo hacían sus operaciones de arriba a abajo, luego si hacías que una celda dependiera de otra que estaba abajo, y repetías este paso, tenías que recalcular todo varias veces para que los nuevos valores se extendieran hacia arriba. (Tenías que disponer de elementos para que dependieran de las celdas superiores.)

Entonces alguien cayó en la cuenta, ¿por qué no realizo las operaciones de cálculo de modo que cada elemento se calcule en función del elemento del que depende? Este algoritmo se llama clasificación topológica. La primera referencia que encontré es de 1963. La patente cubría varias docenas de maneras de implementar la clasificación topológica.

Sin embargo, no conseguirías encontrar esta patente con la búsqueda «hojas de cálculo». No la conseguirías encontrar con la búsqueda «orden natural» ni con la búsqueda «modelo topológico». No incluía ninguno de esos términos. De hecho, estaba descrita como un método para «recopilar fórmulas en código máquina». Cuando la vi por primera vez, pensé que era una patente equivocada.

Supongamos que tienes una lista de patentes y quieres ver qué es lo que no se te permite. Cuando intentas estudiar estas patentes, descubres que son muy difíciles de entender, dado que están escritas en un retorcido lenguaje legal cuyo significado es muy difícil de comprender. Lo que dicen las oficinas de patentes a menudo no significa lo que parece que dicen.

En un estudio del gobierno australiano sobre el sistema de patentes en la década de 1980, se concluía que, aparte de la presión internacional, no había motivos para tener un sistema de patentes —ya que no producía nada bueno para el público— y recomendaba su abolición a pesar de la presión internacional. Una de las cosas que citaban era que los ingenieros no intentan leer las patentes para aprender, porque resulta muy difícil entenderlas. Citaban a un ingeniero que decía: «No puedo reconocer mis propios inventos en las patentes».

No se trata de un asunto meramente teórico. Hacia 1990, un programador llamado Paul Heckel demandó a Apple, alegando que Hypercard infringía dos de sus patentes. Cuando vio Hypercard por primera vez, no pensó que tuviera nada que ver con sus patentes, con sus «invenciones». No se parecía. Cuando su abogado le dijo que se podía interpretar que las patentes se aplicaban a una parte de Hypercard, decidió atacar a Apple. Cuando di una charla sobre esto en Stanford, él estaba entre el público. Dijo, «eso no es verdad, ¡simplemente yo no entendía el alcance de mi protección!». Yo contesté, «sí, eso es lo que yo estaba diciendo».

Así que, en realidad, tendrías que dedicar mucho tiempo a hablar con abogados para hacerte una idea de lo que estas patentes te prohíben hacer. Al final dirán algo como esto: «Si haces algo aquí, seguro que pierdes; si haces algo aquí —Stallman gesticula, señalando una amplia área— hay una posibilidad considerable de que pierdas, y si de verdad quieres estar a salvo, quédate fuera de esta área —vuelve a gesticular, señalando una área todavía más amplia—. Y, por cierto, hay una considerable posibilidad de que como resultado se de curso a una demanda».

Ahora que hemos definido un escenario previsible para hacer negocios, ¿qué vas a hacer? Bien, hay tres posibilidades que podrías probar, cualquiera de las cuales es aplicable en algunos casos.

- Evitar la patente
- Obtener la licencia de la patente

- Revocar la patente en un juicio

Permitidme que describa estas tres posibilidades y qué las hace viables o inviables.

Evitar la patente

«Evitar la patente» —lo que significa no utilizar la idea que cubre la patente. Esto puede ser fácil o difícil, dependiendo de qué idea se trate.

En algunos casos, se patenta una prestación. De este modo, la patente se evita no implementando esa prestación. Por lo tanto sólo cuenta lo importante que sea esa prestación. En algunos casos, puedes prescindir de ella. Hace algún tiempo, los usuarios del procesador de textos XyWrite vieron rebajadas sus aspiraciones. Esa degradación eliminó una prestación que te permitía predefinir abreviaturas. Es decir, cuando escribías una abreviatura seguida de un signo de puntuación, se reemplazaba inmediatamente con alguna prolongación de la abreviatura. Así, podías definir la abreviatura para alguna frase larga, escribirla y entonces la frase aparecía en tu documento. [Los desarrolladores] me escribieron acerca de esto, porque sabían que el editor de Emacs tenía una prestación similar. De hecho, la tenía desde la década de 1970. Este asunto era interesante porque demostraba que había tenido por lo menos una idea patentable en mi vida. ¡Sé que era patentable porque otro la patentó después!

En realidad tomaron en cuenta las tres posibilidades. Primero intentaron negociar con el dueño de la patente y resultó que no negociaba de buena fe. Después pensaron si podrían tener alguna oportunidad de invalidar la patente. Lo que decidieron fue eliminar esa prestación del programa.

Puedes prescindir de ella. Si el procesador de texto sólo carece de esta prestación, quizá la gente todavía lo use. Pero según empiecen a caer varias prestaciones, finalmente te encontrarás con un programa sobre él que la gente piensa que no es muy bueno y tenderá a rechazarlo.

En este caso se trata de una patente bastante limitada sobre una prestación muy específica. Pero, ¿qué se puede hacer en relación a la patente de la British Telecom sobre navegación por hipervínculos por medio del acceso telefónico? Hoy en día, la navegación por hipervínculos es absolutamente esencial para la mayor parte de los usos de los ordenadores. El acceso telefónico también es esencial. ¿Cómo te las arreglas sin esta prestación? La cual, por cierto, ni siquiera es una prestación —realmente es la combinación de dos prestaciones yuxtapuestas de forma arbitraria. Es como tener una patente sobre un sofá y un televisor que están en la misma habitación.

A veces la idea patentada será tan amplia y básica que prácticamente abarca todo un campo; por ejemplo, la idea de clave de uso público, que fue patentada en los EE.UU. La patente caducó en 1997. Hasta entonces, coartó en gran medida el uso de la clave de uso público en EE.UU. Gran cantidad de programas que la gente empezó a desarrollar fueron aplastados —nunca estuvieron de verdad disponibles porque los dueños de las patentes les amenazaban. Posteriormente, se consiguió publicar un programa, el PGP, que inicialmente se lanzó como software libre. Al parecer, cuando los dueños de las

patentes estaban a punto de atacar, se dieron cuenta de que eso podría hacerles muy mala publicidad. Así que impusieron restricciones, haciendo que sólo fuera para uso no comercial, lo que significaba que no podría hacerse muy popular. De este modo limitaron el uso de la clave de uso público por una década o más. No había otras vías alternativas a esa patente. No había nada que pudieras hacer y fuera comparable a la clave de uso público.

A veces se patenta un determinado algoritmo. Por ejemplo, existe una patente sobre una versión mejorada de Fast Fourier Transform (FFT). Funciona dos veces más rápido, más o menos. Puedes evitarlo usando un FFT normal en tu programa. Esta parte del programa tarda el doble. Quizás eso no importe, quizás represente una pequeña parte del tiempo de carga del programa. Quizás si es dos veces más lento, ni siquiera te des cuenta. O quizás tu programa no funcione en absoluto dado que le llevará el doble de tiempo hacer su trabajo. Los efectos difieren.

En algunos casos, puedes encontrar un algoritmo mejor. Esto podría ser bueno o no. Como en el proyecto GNU no podíamos usar Compress, empezamos a buscar un algoritmo alternativo para la compresión de datos. Alguien nos escribió diciendo que tenía uno; había escrito un programa y quería ofrecérnoslo. Íbamos a sacarlo. Por casualidad, coincidió que leí un ejemplar del New York Times. (No leía un ejemplar del Times más que una vez cada pocos meses.) Así que le eché un vistazo y decía que alguien había recibido una patente por «inventar un nuevo método de compresión de datos». Supuse que sería mejor revisar esa patente. Conseguí una copia y resultó que cubría el programa que íbamos a lanzar en una semana. Ese programa murió antes de nacer.

Más tarde encontramos otro algoritmo que no estaba patentado. Éste llegó a ser el programa gzip, que ahora es de hecho el estándar para la compresión de datos. Como algoritmo para usar en un programa de compresión de datos, estaba bien. Cualquiera que quisiera hacer compresión de datos podría usar gzip en lugar de Compress.

La misma patente de compresión basada en el algoritmo de LZW también fue usada en formatos de imagen como GIF. Pero en este caso, dado que el trabajo que la gente quería hacer no era simplemente comprimir datos sino construir una imagen que pudieran activar con su software, resultó muy difícil comenzar a usar un algoritmo diferente. ¡No hemos sido capaces de hacerlo en diez años! Sí, la gente usaba el algoritmo de gzip para definir otro formato de imagen una vez que empezaron a ser amenazados con demandas por usar archivos de GIF. Cuando empezamos a decirle a la gente que dejara de usar archivos de GIF, que se cambiara, la gente decía «no podemos cambiarnos, los navegadores todavía no admiten el nuevo formato». Los desarrolladores de navegadores decían «esto no nos agobia, después de todo, nadie está usando este nuevo formato de archivo».

En efecto, existía mucha inercia en la sociedad en el uso del formato GIF, no fuimos capaces de que la gente se cambiara. En esencia, el uso que hace la comunidad del formato GIF todavía apremia a los sitios web a usar el formato GIF, con el resultado de que son vulnerables a estas amenazas.

De hecho, la situación es todavía más extraña. En realidad hay dos patentes que cubren el algoritmo de compresión de LZW. La oficina de patentes ni siquiera podía

decir que estaban publicando dos patentes sobre la misma cosa; no podían seguir el rastro. Hay un motivo para esto: hace falta dedicación al estudio de estas dos patentes para darte cuenta de que realmente protegen la misma cosa.

Si fueran patentes sobre procesos químicos, sería mucho más fácil. Podrías comprobar qué sustancias se están usando, qué entradas y salidas hay, que acciones físicas se están tomando. Sin importar cómo estuvieran descritas, comprobarías qué son y comprobarías que son parecidas. Si algo es puramente matemático, existen muchas maneras muy diferentes de describirlo. A primera vista no parecen similares. Tienes que comprenderlos de verdad para comprobar que realmente están hablando de la misma cosa. La oficina de patentes no tiene tiempo. Hace unos pocos años, la oficina de patentes de los EE.UU. estaba empleando una media de 17 horas por patente. Esto no es mucho tiempo para pensar en ellas con cuidado, así que por supuesto comenten errores como éste. De hecho, os he hablado de un programa que murió antes de nacer. Ese algoritmo también disponía de dos patentes en los EE.UU.; por lo que parece, no es algo tan raro.

Evitar las patentes puede ser fácil, o puede ser imposible. Podría ser fácil y hacer inútil vuestro programa —según la situación.

Aquí llegamos a otro punto que se debería mencionar. A veces una empresa o consorcio puede hacer que un formato o un protocolo sea *de facto* el estándar. Por lo tanto, si se patenta ese formato o protocolo es un auténtico desastre. Incluso, existen estándares oficiales que están restringidos por patentes. Se produjo un gran alboroto político en septiembre de 2001 cuando el W3C (Consorcio de la World Wide Web) propuso que se empezaran a adoptar estándares protegidos por patentes. La comunidad protestó, y tuvieron que desdecirse. Volvieron a insistir en que cualquier patente debería ser implementada libremente por cualquiera y en que los estándares tenían que ser libres para que cualquiera los implementara. Es una victoria interesante. Pienso que esta fue la primera vez que una organización de estándares ha tomado esta decisión. Es normal que las organizaciones de estándares deseen añadir algo restringido por las patentes a un estándar para que a la gente no se le permita implementarlo libremente. Tenemos que acudir a otras organizaciones de estándares y reclamarles que cambien sus reglas.

Obtener la licencia de la patente

La segunda posibilidad consiste en conseguir una licencia para la patente en lugar de evitar la patente. Esta no es necesariamente una opción. El dueño de la patente no tiene por qué ofrecerte la licencia, no es obligatorio. Hace diez años, la Liga para la Libertad de Programación recibió una carta pidiendo ayuda para alguien cuyo pequeño negocio estaba fabricando máquinas tragaperras para los casinos, que ya entonces usaban ordenadores. Este alguien recibió una amenaza de otra empresa que decía: «Tenemos una patente. No se os permite hacer esto. ¡Cerrad!».

Le eché un vistazo a esa patente. Cubría la tenencia de una serie de ordenadores en red para instalar juegos, de modo que cada ordenador proveía más de un juego y te permitía jugar a más de un juego a la vez.

Os parecerá que la oficina de patentes de verdad piensa que es algo brillante hacer cualquier cosa más de una vez. No se dan cuenta de que en la ciencia informática esta es la forma más obvia de generalizar algo. Lo hiciste una vez, luego ahora lo puedes hacer varias veces, puedes crear una subrutina. Piensan que si haces algo más de una vez, de algún modo significa que eres brillante, que posiblemente nadie pueda discutir contigo y que tienes derecho a mandar.

De todos modos, a esta persona no le ofrecieron la licencia. Tuvo que cerrar. Ni siquiera podía permitirse ir a juicio. Yo diría que esa patente en concreto era una idea obvia. Es posible que un juez hubiera estado de acuerdo, pero nunca lo sabremos porque esta persona no se podía permitirse ir a juicio.

Sin embargo, muchos dueños de patentes ofrecen licencias. Aunque a menudo cobran mucho dinero por ello. La compañía que daba licencias para la patente del cálculo en orden natural pedía un cinco por ciento de los ingresos brutos por cada hoja de cálculo vendida en los EE.UU. Me han dicho que ese era el precio barato, anterior a la demanda —si de verdad te demandaban y ganaban, te exigían más.

Quizás puedas permitirte ese cinco por ciento por la licencia de esa patente, pero ¿y si necesitas la licencia de veinte patentes para desarrollar el programa? La gente del sector me dijo que, prácticamente, dos o tres licencias como esa harían inviable cualquier negocio.

Existe una situación en la que obtener una licencia por el uso de la patente es una solución muy buena. Es lo que ocurre si eres una megacorporación multinacional. Puesto que estas empresas poseen muchas patentes y se intercambian las licencias entre ellas, se libran de gran parte del daño que el sistema de patentes provoca y sólo perciben las cosas buenas.

IBM publicó un artículo en la revista Think —creo que era el número cinco de 1990— sobre el catálogo de patentes de IBM, en éste se exponía que IBM percibía dos tipos de beneficios en concepto de sus 9.000 patentes en EE.UU. —creo que el número es mayor hoy en día. Estos eran, en primer lugar, los ingresos por royalties, y en segundo lugar, «el acceso a las patentes de otros». Decían que el segundo beneficio era de mayor magnitud. De tal forma que el beneficio que IBM percibía por tener permiso de usar las ideas patentadas por otros era diez veces el beneficio directo que IBM percibía por ofrecer licencias.

¿Qué significa esto realmente? ¿Qué beneficio percibe IBM de su «acceso a las patentes de otros»? Esencialmente es el beneficio de estar exento de los problemas que el sistema de patentes puede causarle. El sistema de patentes es como la lotería: lo que ocurre con una patente determinada puede no ser nada, puede ser un golpe de suerte para algún dueño de una patente o un desastre para todos los demás. Pero IBM es una empresa demasiado grande, le compensa. Ellos pueden estimar el promedio de ventajas y desventajas del sistema de patentes. Para ellos, los problemas del sistema de patentes podrían haber sido diez veces mayores que las ventajas.

Digo «podrían haber sido» porque a través del intercambio de patentes se evitan experimentar esos problemas. Esos problemas sólo son potenciales, en realidad no les

afectan. Pero cuando miden los beneficios de evitarlos, lo estiman en diez veces el valor del dinero que ingresan por sus patentes.

Este fenómeno del intercambio de licencias desmiente un mito común, el mito del «genio famélico», el mito de que las patentes «protegen» al «pequeño inventor». (Son términos propagandísticos. No deberíais usarlos.)

La historia es como sigue: imagina que existe un «brillante» diseñador de lo que sea. Imagina que se ha pasado «años de privaciones en el desván» diseñando un nuevo y maravilloso prototipo y ahora quiere fabricarlo. ¿No es una vergüenza que las grandes empresas vayan a competir con él, que se queden con todo el negocio y él «pase hambre»?

Debo precisar que normalmente aquellos que trabajan en el sector de las tecnologías de vanguardia no trabaja por su cuenta, que las ideas no salen de la nada —están basadas en las ideas de otros— y que, hoy por hoy, esta gente tiene muy buenas oportunidades de conseguir un trabajo si lo necesita. Así que este cuento —la idea de que una idea brillante venga una persona que trabaja sola— no es realista, al igual que la idea de que se encuentre en riesgo de pasar hambre.

Pero sí se puede concebir que alguien tenga una idea y que esta idea junto con otras 100 o 200 ideas pueda ser la base para la fabricación de algún tipo de producto, y que las grandes compañías podrían querer competir con esta persona. Así que veamos qué pasa si esa persona intenta usar una patente para impedírselo. Él dice: «Ah, no, IBM, no puedes competir conmigo. Tengo esta patente». IBM dice: «Veamos. Echemos un vistazo a tu producto. Hmmm. Tengo esta patente, y esta otra, y esta otra y esta otra y esta otra y esta otra, que han sido violadas por algunas partes de tu producto. Si crees que puedes luchar contra todas ellas en un juicio, volveré y encontraré unas cuantas más. Así que, ¿por qué no intercambias tus licencias con las mías?». Y entonces el brillante pequeño inventor dice, «bueno, vale, las intercambio». Entonces puede volver y fabricar este maravilloso lo-que-sea, pero también puede hacerlo IBM. IBM obtiene «acceso» a su patente y el derecho a competir con él, lo que quiere decir que esta patente no le «protegió» en absoluto. El sistema de patentes no hace eso, en realidad.

Las megacorporaciones evitan, en su mayoría, el daño del sistema de patentes; principalmente ven la cara buena. Por eso quieren tener patentes de software: son las únicas que se beneficiarán de ello. Pero si eres un pequeño inventor o trabajas para una pequeña empresa, la pequeña empresa no será capaz de hacer esto. Lo intentan. El problema es que las pequeñas empresas no pueden conseguir suficientes patentes para hacer que todo el mundo intercambie sus licencias con ellas.

Cualquier patente apunta a una cierta dirección. De modo que si una pequeña empresa tiene patentes que apuntan allí, y allí, y allí, y alguien por allí [Stallman señala a otro sitio] les señala un patente y dice dame tu dinero, la empresa pequeña está desamparada. IBM puede hacerlo, porque con 9.000 patentes apuntan a todas partes, no importa dónde estés, probablemente haya una patente de IBM que te señale. Así que IBM casi siempre puede hacerte intercambiar la licencia. Las empresas pequeñas ocasionalmente pueden hacer que alguien les intercambie las suyas. Dirán que quieren las

patentes para fines defensivos, pero no conseguirán las suficientes para defenderse a sí mismas.

Hay casos en que ni siquiera IBM puede hacer que alguien le intercambie sus licencias. Esto ocurre cuando hay una compañía cuyo único negocio es tomar una patente y exprimirle a la gente dinero. La empresa que tenía la patente del orden natural de cálculo era exactamente este tipo de empresa. Su único negocio era amenazar a la gente con una demanda e ingresar dinero de gente que estaba creando algo de verdad.

No hay patentes sobre los procedimientos legales. Supongo que los abogados comprenden qué lata sería tener que tratar ellos mismos con el sistema de patentes. El resultado es que no hay forma de obtener una patente para hacer que tal compañía intercambie sus licencias contigo. Así que van por ahí exprimiendo a todo el mundo. Pero supongo que empresas como IBM se imaginan que es parte del precio de hacer negocios, así que pueden vivir con ello.

Así que esta es la opción de obtener una licencia de patente, que puede ser posible o no, según seas capaz de permitírtelo o no —lo cual nos conduce a la tercera posibilidad.

Revocar la patente en un juicio

Supuestamente, para que algo sea patentado, tiene que ser nuevo, útil y no obvio. (Es el léxico usado en EE.UU., creo que otros países tienen otro bastante similar.) Por supuesto, cuando la oficina de patentes entra en juego, comienza por interpretar «nuevo» y «no obvio». «Nuevo» viene a significar «no lo tenemos en nuestros archivos» y «no obvio» tiende a significar «no obvio para alguien con un coeficiente intelectual de 50».

Un estudioso de la mayoría de las patentes de software publicadas en EE.UU. —o que al menos lo era, no sé si todavía puede con todas ellas— dijo que el 90 por ciento no habría pasado el «test de Cristal City»,[2]lo que quería decir que si el personal de la oficina de patentes bajara al kiosko y adquiriera algunas revistas de informática, comprobaría que esas ideas ya son conocidas.

La oficina de patentes hace cosas que son tan obviamente estúpidas, que ni siquiera tendrías que conocer el estado de la técnica para saber que son estúpidas. Esto no se limita al software. Una vez vi el famoso ratón patentado de Harvard, que fue obtenido después de que Harvard practicara ingeniería genética con un ratón introduciéndole un gen cancerígeno. El gen cancerígeno ya era conocido y fue insertado usando técnicas conocidas en una variedad ya conocida de ratón. La patente que obtuvieron cubría la introducción de cualquier gen cancerígeno dentro de cualquier tipo de mamífero mediante cualquier método. No tienes que saber nada sobre ingeniería genética para darte cuenta de que esto es ridículo. Me han dicho que esta «sobrepretensión» es una práctica normal, y que la oficina de patentes de los EE.UU. a veces invita a los solicitantes a hacer sus pretensiones todavía más amplias. Esencialmente, uno tiende a hacer tan amplias

[2]Cristal City es la zona de Washington donde se encuentra la oficina de patentes. [*N. del E.*]

las pretensiones hasta que cree que está en contradicción con algo que no es ambiguo y está bien documentado. Observad cuanta tierra podéis conseguir en el espacio mental.

Cuando los programadores echan un vistazo a muchas patentes de software dicen: «¡Esto es ridículamente obvio!». Los burócratas de las patentes tienen todo tipo de excusas para justificar su ignorancia sobre lo que piensan los programadores. Dicen: «¡Ah!, pero tienes que considerarlo en términos de cómo eran las cosas hace diez o veinte años». En ese momento descubrieron que si repiten algo hasta la saciedad pueden hacerte perder toda perspectiva. Todo puede parecer no obvio si lo descompones y lo analizas suficientemente. Uno simplemente pierde toda criterio de obviedad o por lo menos pierde la habilidad de justificar cualquier categoría sobre lo obvio o no. Luego, por supuesto, describen a los dueños de las patentes como brillantes inventores, sin excepción, por eso no podemos cuestionar su derecho a tener poder sobre lo que hacemos.

Si vas a juicio, los jueces tienden a ser un poco más estrictos sobre qué es obvio y qué no. El problema es que cuesta millones de dólares.

Una vez oí hablar de un caso sobre patentes, recuerdo que la parte demandada era Qualcomm, y creo que el fallo finalmente fue de 13 millones, de los cuales la mayoría se destinó a pagar a los abogados de las dos partes. Quedaron unos pocos millones de dólares para el demandante —porque Qualcomm perdió.

En gran medida, la validez de una patente dependerá de las incidencias en el historial. Muchas incidencias en el historial como, exactamente qué fue publicado y cuándo, y cuáles de esos elementos se pueden encontrar, cuáles no se perdieron, fechas concretas y así... determinan si una patente es válida o no.

En realidad, resulta extraño que la patente de British Telecom «seguimiento de hipervínculos por medio de acceso telefónico» fuera solicitada en 1975. Creo que fue en 1974 cuando creé el paquete Info por primera vez. El paquete Info te permite utilizar hipervínculos y la gente usaba el teléfono para conectarse y acceder al sistema. Así que de hecho, yo produje una prueba que invalidaba esta patente. Es la segunda idea patentable que sé que he producido en mi vida.

Pero no creo que tenga ninguna prueba de ello. No pensé que esto fuera lo suficientemente interesante como para publicarlo. Después de todo, la idea de seguir un hipervínculo la tomé de la demo de Englebart para su editor. Él fue quien tuvo una idea que era interesante de publicación. Lo que yo había hecho lo llamé «el hipertexto del pobre» dado que tenía que implementarlo en el contexto de TECO. No tenía tanta capacidad como su hipertexto, pero al menos era útil para buscar documentación, que era todo lo que pretendía. Y en cuanto a que hubiera acceso telefónico al sistema, bueno, lo había, pero no se me ocurrió que lo uno tuviera nada que ver con lo otro. No iba a publicar un artículo que dijera: «¡Oh, he implementado el hipertexto del pobre y ¿sabéis qué? ¡También hay líneas telefónicas en el ordenador!».

Sospecho que no hay manera de precisar en qué fecha implementé esto. ¿Fue publicado de algún modo? Bueno, invitamos a la gente a que entrara a través de ARPANET, y se registrara en nuestra máquina —de modo que hubieran podido buscar documentación a través de Info y echar un vistazo al asunto. Si nos lo hubieran preguntado, se

habrían encontrado con que teníamos acceso telefónico. Como podéis ver, las incidencias en el historial determinan si tienes una técnica original.

Ahora, por supuesto, hay una publicación hecha por Englebart sobre el hipertexto, que ellos, lo acusados, van a mostrar. De todos modos, no creo que diga nada sobre tener líneas telefónicas en el ordenador así que no está claro si bastará.

La posibilidad de ir a juicio para revocar una patente es una opción. Debido a los gastos normalmente ni se plantea, aunque puedas encontrar una prueba sólida que sea suficiente para revocar la patente. Como resultado, una patente nula, una patente que nominalmente no debería haber existido —pero en realidad muchas de estas patentes sí existen— es un arma peligrosa. Si alguien te ataca con una patente nula, verdaderamente te puede causar muchos problemas. Puedes ir de farol enseñándoles tus pruebas. Depende de si se pueden asustar de este modo o no. Podrían pensar: «Bueno, vas de farol, nos imaginamos que realmente no puedes ir a juicio: no te lo puedes permitir, así que de todos modos te demandaremos».

Todas estas opciones son cuestiones con las que a veces te puedes apañar, pero con las que a menudo no puedes. Así que tendrás que enfrentarte a patente, tras patente, tras patente. Cada vez que seas capaz de encontrar alguna de estas tres posibilidades, te encuentras con que existe otra patente, luego otra y luego otra. Se convierte en algo parecido a cruzar un campo de minas. Cada paso que das, cada decisión de diseño posiblemente no pise una patente, así que puedes dar unos pocos pasos y posiblemente no habrá una explosión. Pero las posibilidades de que te abras paso a través del campo de minas y crees el programa que quieres desarrollar sin pisar nunca una patente disminuyen más y más según el programa se hace más grande.

Bueno, la gente solía decirme, «bien, hay patentes en otros campos, ¿por qué el software debería estar eximido de ellas?» Observad qué suposición más grotesca tenemos aquí: de algún modo todos debemos sufrir por el sistema de patentes. Es como decir: «Alguna gente desarrolla cáncer ¿por qué tú deberías estar exento?» Tal y como yo lo veo, que una persona no desarrolle cáncer es algo bueno.

Pero detrás de eso hay una pregunta menos sesgada, una buena pregunta, que es: ¿es el software diferente de los demás campos? ¿Debería la política de patentes ser diferente en campos diferentes? ¿En ese caso, por qué?

Permitidme que conteste a esa pregunta: las patentes se relacionan con diferentes campos de forma diferente porque, en campos diferentes, las patentes se relacionan con los productos de forma diferente.

De un extremo tenemos a las empresas farmacéuticas, donde una fórmula dada podría patentarse de modo que esa patente cubra un solo producto. Una sustancia nueva no estaría protegida por la patente que ya existe. De existir una patente para este nuevo producto, sería el dueño de la patente quien desarrollaría el nuevo producto.

Eso encaja con la idea ingenua que tenemos del sistema de patentes, que si estas diseñando un producto nuevo, vas a conseguir «la patente». La idea es que hay una patente por producto que cubre la idea del producto. En algunos campos esto está cerca de ser verdad; en otros campos esto está lejos de ser verdad.

El campo del software está en el último extremo: un programa puede ser objeto de muchas patentes. Esto pasa porque los paquetes de software son normalmente muy grandes. Usan muchas ideas diferentes en combinación. Si el programa es nuevo y no es una simple copia, entonces probablemente está usando una combinación diferente de ideas. Por supuesto, incorporadas en un código escrito de nuevo, porque no puedes nombrar mágicamente estas ideas y hacerlas funcionar. Tienes que implementar todas. Tienes que implementar todas ellas en esa combinación.

El resultado es que incluso cuando escribes un programa, estás usando una enorme cantidad de ideas diferentes, cada una de las cuales puede estar patentada por alguien. Un par de ellas podría estar patentada por alguien en una combinación. Podría haber varias maneras distintas de describir una idea, que podrían estar patentadas por gente distinta. Así que posiblemente hay miles de cosas, miles de puntos vulnerables en tu programa, que podrían estar ya patentadas por cualquier otro.

Por eso las patentes de software tienden a obstruir el progreso del software —el trabajo de creación de software. Si fuera el caso de «un producto, una patente» entonces estas patentes no obstruirían la creación de productos porque al crear un nuevo producto, no habría manera de que estuviera patentado por nadie más. Pero cuando un producto se corresponde con muchas ideas diferentes combinadas, parece probable que tu nuevo producto —tanto en parte como en su totalidad— pueda estar patentado ya por otro.

En realidad, hay investigaciones económicas que demuestran cómo la imposición de un sistema de patentes, en un campo en el que existe una creciente innovación, puede ralentizar el progreso. Los defensores de las patentes de software dicen: «Bueno, sí, a lo mejor hay problemas, pero más importante que cualquier problema, es el hecho de que las patentes deben promover la innovación, y esto es tan importante que da igual los problemas que causen». Por supuesto, esto no lo dicen en voz alta porque sería ridículo, pero implícitamente quieren haceros creer que el sistema de patentes promueve siempre el progreso y que eso compensa todo coste posible. Pero en realidad no hay razones para creer que promueva el progreso. Ahora tenemos un modelo que demuestra exactamente cómo las patentes pueden ralentizar el progreso. El caso donde ese modelo aplicado describe muy bien el software es el de la creciente innovación.

¿Por qué se encuentra el software en ese extremo del espectro? El motivo es que en software creamos objetos matemáticos ideales. Puedes construir un castillo enrevesado que descanse sobre una línea fina y se mantendrá en pie porque no pesa nada. En otros campos, la gente se tiene que manejar con la obstinación de la materia —la de los objetos físicos. La materia hace lo que tiene que hacer. Puedes intentar modelarla, pero si el comportamiento real no se ajusta a tu modelo entonces peor para ti, porque el desafío es hacer objetos físicos que verdaderamente funcionen.

Si quiero poner una orden condicional en una orden «mientras», no me tengo que preocupar sin la condicional oscilará a tal frecuencia y se frotará contra el «mientras» hasta que finalmente ambas se rompan. No me tengo que preocupar de si oscilará a una determinada alta frecuencia y provocará una señal en el valor de otra variable. No me tengo que preocupar de cuánta corriente atraerá esa condicional, ni de si disipará el

calor dentro del «mientras», o de si habrá una caída en el voltaje a través del «mientras» que hará que la orden condicional no funcione. No me tengo que preocupar de que si pongo este programa en un entorno de agua salada, el agua salada se podría introducir entre la condicional y la orden «mientras» y provocar corrosión. [Risas del público.]

No me tengo que preocupar, cuando me refiero al valor de una variable, de si estoy excediendo su aforo refiriéndome a ella 20 veces. No me tengo que preocupar de cuánta capacidad tiene, ni de si habrá suficiente tiempo para que cobre valor.

No me tengo que preocupar, cuando escribo el programa, de cómo voy a juntar físicamente cada copia ni de si puedo arreglármelas para llegar a poner la condicional dentro del «mientras». No me tengo que preocupar de cómo voy a acceder a ella, en caso de que la orden condicional se rompa, para retirarla y cambiarla por una nueva. Hay muchos problemas de los que no nos tenemos que preocupar en el software; eso hace mucho más fácil escribir un programa que diseñar un objeto físico que tenga que funcionar.

Esto puede parecer extraño, porque habréis oído a gente hablando sobre lo difícil que es diseñar software, sobre el enorme problema que supone y reflexionando sobre cómo van a resolverlo. Realmente no están hablando de lo mismo que yo. Yo estoy comparando sistemas físicos y sistemas de software de la misma complejidad, con la misma cantidad de elementos. Estoy diciendo que un sistema de software es mucho más fácil de diseñar que un sistema físico. Pero el talento de la gente en estos campos diferentes es el mismo, de modo que ¿qué hacemos cuando nos encontramos con un campo fácil? ¡Lo hacemos avanzar! Llevamos nuestras habilidades al límite. Si los sistemas del mismo tamaño son fáciles, hagamos sistemas diez veces más grandes —¡resultará entonces más difícil! Eso es lo que hacemos: producimos sistemas de software que son mucho más grandes que los sistemas físicos en cuanto a su número de elementos.

Un sistema físico cuyo diseño incluye un millón de partes diferentes es un megaproyecto. Un programa informático cuyo diseño incluye un millón de partes quizá tenga 300.000 líneas; unas pocas personas escriben eso en un par de años. No es un programa especialmente gigantesco. GNU EMACS tiene ahora varios millones de partes en su diseño, creo. Tiene un millón de líneas de código. Este es un proyecto esencialmente hecho sin financiación de ningún tipo, realizado en su mayoría por gente en su tiempo libre.

Hay otra gran salvedad. Si has diseñado un producto físico, lo próximo que debes hacer es diseñar la fábrica para producirlo. Construir esta fábrica podría costar millones o decenas de millones, mientras que para hacer copias de un programa sólo tienes que pulsar «copiar». El mismo comando copiará cualquier programa. Si quieres copias en un CD, perfecto, grabas un CD master y lo envías a una planta de CDs. Ellos usarán el mismo equipo que copia cualquier contenido en un CD. No hace falta construir una fábrica especializada para producir cada producto concreto. Se da una tremenda simplificación y reducción del coste del diseño.

Una compañía automovilística, que se gasta 50 millones para construir una fábrica para hacer un nuevo modelo de coche, puede contratar a unos abogados para vérselas con las negociaciones de la licencia de patentes. Incluso pueden vérselas con una

demanda si quisieran. Diseñar un programa de la misma complejidad podría costar 50.000 o 100.000 dólares. En comparación, el coste de tratar con el sistema de patentes es demoledor —en realidad diseñar un programa de la misma complejidad que el diseño mecánico de un coche representa probablemente un mes de trabajo. Cuántas partes tiene un coche. . . , quiero decir, en caso de que el coche no tenga ordenador.[3] Esto no quiere decir que diseñar uno bueno sea fácil, sólo que no incluye tantos elementos diferentes.

El resultado es que el software es realmente distinto de otros campos, porque cuando estamos trabajando con herramientas matemáticas, diseñar algo es mucho, mucho más fácil. El resultado es que producimos con regularidad sistemas que son mucho, mucho más grandes y lo hacemos con sólo unas pocas personas. El resultado es que en lugar de acercarnos a tener un producto y una patente, estamos en un sistema donde un producto implica muchas, muchas ideas que podrían estar ya patentadas.

El mejor modo de explicar esto por analogía es con las sinfonías. Una sinfonía también es larga y incluye muchas notas, y probablemente usa muchas ideas musicales. Imaginad que los gobiernos europeos del siglo XVIII hubieran decidido que querían promover el progreso de la música sinfónica estableciendo una Oficina Europea de Patentes Musicales, que ofreciera patentes para cualquier tipo de ideas musicales que pudieras exponer con palabras.

Imaginad entonces que estamos cerca de 1800, que sois Beethoven y queréis escribir una sinfonía. Os encontraréis con que disponer vuestra sinfonía de modo que no infrinja ninguna patente es más difícil que escribir una buena sinfonía.

Cuando os quejáis de esto, los dueños de las patentes os dicen «Venga, Beethoven, ya nos estás jodiendo porque no tienes ideas propias. Lo único que quieres es robar nuestras invenciones». Beethoven, como de hecho sucedía, tenía muchas ideas musicales nuevas —pero tenía que usar muchas ideas musicales ya existentes para hacer música reconocible, para hacer música que pudiera gustar a los oyentes, que estos pudieran reconocer como música. Nadie es tan brillante como para reinventar una música completamente distinta y hacer algo que a la gente le guste escuchar. Pierre Boulez dijo intentar hacerlo, pero ¿quién escucha a Pierre Boulez?

Nadie es tan brillante como para reinventar toda la ciencia informática, de forma completamente nueva. Si lo hiciera, produciría algo que los usuarios encontrarían tan extraño que no querrían usarlo. Si hoy echas un vistazo a un procesador de texto, te encontrarás, creo, con cientos de características diferentes. Si desarrollas un procesador de texto nuevo, bueno e innovador, eso significa que incluye algunas ideas nuevas, pero debe incluir cientos de viejas ideas. Si no te está permitido usarlas, no puedes hacer un procesador de textos innovador. Como el trabajo de creación de software es tan grande, el resultado es que no necesitamos ningún plan artificial para incentivar nuevas ideas.

---

[3]Hay aproximadamente 300-400 elementos distintos en una transmisión automática y una transmisión es generalmente el componente más complicado de un coche. Diseñar una transmisión puede llevar de seis meses a un año, e incluso entonces puede llevar más tiempo tenerla a punto y en funcionamiento. De todos modos, un programa con 500 o 800 elementos útiles tendría entre 200 y 300 líneas de código, y probablemente a un buen programador le llevaría de un día a una semana escribirlo, probarlo y depurarlo.

Simplemente deja que la gente escriba software y ya tendrán nuevas ideas. Si quieres escribir un programa y quieres hacerlo bien, te vendrán a la cabeza algunas ideas y encontrarás alguna forma de usar algunas de ellas.

Lo que solía pasar —porque yo estaba en el campo del software antes de que existieran patentes de software— era que la mayoría de los creadores publicaban cualquier idea que pensaran que fuera digna de atención y por la que pensaran recibir algún reconocimiento o respeto. Las ideas que fueran demasiado pequeñas o no lo suficientemente notables no las publicaban porque podrían ser una tontería. Ahora se supone que el sistema de patentes apoya el descubrimiento de ideas. En realidad, en los viejos tiempos, nadie guardaba las ideas en secreto. Guardaban el código en secreto, es verdad. El código, después de todo representaba el grueso del trabajo. Guardaban el código en secreto y publicaban las ideas, de modo que los empleados adquirieran algo de reconocimiento y se sintieran bien.

Después de las patentes de software, todavía guardan el código en secreto y además patentan las ideas, así que en realidad, no se ha apoyado el descubrimiento en ningún sentido significativo. Las mismas cosas se guardan en secreto hoy al igual que se guardaban en secreto ayer, pero las ideas que se solían publicar para que las pudiéramos usar, es muy probable que ahora sean patentadas y estén fuera de nuestro alcance durante 20 años.

¿Qué puede hacer un país para cambiar esto? ¿En qué dirección deberíamos modificar las políticas al respecto para solucionar este problema?

Hay dos puntos donde se puede atacar. Uno es el punto desde donde se lanzan las patentes, la oficina de patentes. El segundo es donde se aplican las patentes. Aquí se trata de qué es lo que protege la patente.

Una forma es mantener un buen criterio para publicar patentes. Esto puede funcionar en un territorio que no ha autorizado antes las patentes informáticas, por ejemplo, en la mayor parte de Europa. Simplemente reforzar claramente las reglas de la Oficina Europea de Patentes que dictan que el software no es patentable ya es una buena solución para Europa. Ahora Europa está teniendo en cuenta una directiva sobre patentes de software. (Supongo que la directiva será más amplia, pero una de sus implicaciones importantes son las patentes de software.) Simplemente con modificar esta directiva para dictar que las ideas de software no pueden ser patentadas, el problema se mantendrá alejado de Europa, al menos en su mayor parte, excepto en algunos países que podrían haber asumido el problema por su cuenta, siendo por desgracia el Reino Unido uno de ellos —por desgracia para vosotros.

Esa posibilidad no existe en EE.UU. La razón es que EE.UU. ya tienen una gran cantidad de patentes de software y cualquier cambio en el criterio para publicar patentes no se deshará de las que ya existen.[4] Así, en los EE.UU., la solución tendrá que pasar

[4]Digo «patentes de software», pero ¿a qué me estoy refiriendo? La oficina de patentes de los EE.UU. no divide oficialmente las patentes entre patentes de software y otras patentes. Así que, de hecho, se puede concebir que cualquier patente pueda servir para demandarte si se puede aplicar a algún software. Las patentes de software son patentes que potencialmente se podrían aplicar al software, patentes que en potencia pueden servir para demandarte por escribir software.

por cambiar la aplicabilidad, el alcance, de las patentes. Dictar que una implementación pura de software, instalada sobre hardware de uso general que no infringe en sí mismo la patente, no está protegida por ninguna patente y no puede ser objeto de demanda por ello. Este es otro tipo de solución.

El primer tipo de solución, la solución que interviene sobre qué tipos de patentes pueden ser válidos, es una buena solución para Europa.

Cuando en EE.UU. se empezaron a conceder patentes de software, no hubo debate político. En realidad, nadie se enteró. El ámbito del software, en su mayor parte, ni siquiera se enteró. Existía una decisión del Tribunal Supremo en 1981 que reflexionaba sobre la patente de un procedimiento para curar el síndrome del nevus azul. El fallo fue que el hecho de que el aparato incluyera un ordenador y un programa como parte del procedimiento para curar el síndrome no lo hacía impatentable. Al año siguiente, la sala de apelaciones que considera todos los casos de patentes invirtió los términos: dictó que el hecho de que hubiera un ordenador y un programa en todo esto lo hacía patentable. El hecho de que cualquier cosa tenga un ordenador y un programa la hace patentable. Por eso los EE.UU. empezaron a tener patentes sobre procesos de negocio: porque los procesos de negocio se gestionaban con un ordenador y eso los hacía patentables.

De este modo, se dictó este fallo y creo que la patente sobre el cálculo en orden natural fue una de las primeras o quizá incluso haya sido la primera.

Durante la década de 1980 no sabíamos nada de esto. Fue hacia 1990 cuando los programadores en los EE.UU. empezaron a ser conscientes de que se enfrentaban a un peligro con las patentes de software. Vi cómo trabajaba el sector antes y cómo trabajaba después. No vi ningún aceleramiento especial del progreso después de 1990.

En EE.UU. no hubo debate político pero en Europa, ha habido un gran debate político. Hace varios años hubo presiones para enmendar el tratado de Múnich que establecía la Oficina Europea de Patentes. Tenía una cláusula que dictaba que el software no es patentable. La presión fue para enmendarlo y para que se pudiera empezar a permitir las patentes de software. Sin embargo, la comunidad se enteró. Fueron realmente los desarrolladores y los usuarios de software libre quienes llevaron la iniciativa. Pero no somos los únicos amenazados por las patentes de software. Todos los desarrolladores de software están amenazados por las patentes de software, y incluso los usuarios de software están amenazados por las patentes de software.

Por ejemplo, Paul Heckel —cuando Apple no estaba muy asustada por sus amenazas— amenazó con empezar a demandar a los clientes de Apple. Apple encontró esto mucho más temible. Se imaginaron que no podrían permitirse que sus clientes fueran demandados de este modo, aunque finalmente ganaran. Así que los usuarios también pueden ser demandados, bien como forma de atacar al creador, bien como simple forma de exprimirles dinero por su cuenta o bien como forma de causar el caos. Todos los desarrolladores y usuarios de software son vulnerables.

Sin embargo, fue la comunidad del software libre en Europa la que llevó la iniciativa para organizar la oposición. De hecho, dos tercera partes de los países que están ahora en la Oficina Europea de Patentes votó en contra de enmendar ese tratado. En ese momento, la UE intervino, los directores estaban divididos acerca de este cuestión.

Al parecer, el encargado de la promoción del software está en contra de las patentes de software, pero no estaba a cargo de este asunto. Es la dirección del Mercado Único la que tiene esta competencia y está dirigida por alguien que está a favor de las patentes del software. Básicamente no hicieron caso de la opinión pública que se les había expresado. Han propuesto una directiva para permitir las patentes de software.

El gobierno francés ya ha dicho que está en contra. La gente está informando a otros gobiernos europeos para que se opongan a las patentes del software y es vital que se empiece a hacer esto aquí, en Inglaterra. Según Harmut Pilch, que es uno de los líderes en la lucha europea contra las patentes de software, el mayor impulso para éstas viene de la oficina británica de patentes. La oficina de patentes del Reino Unido está simplemente inclinada a favor de las patentes de software. Hizo una encuesta pública y la mayoría de las respuestas se oponían a las patentes de software. Entonces escribieron un informe diciendo que la gente parecía satisfecha con ellas, pasando por alto completamente las respuestas. La comunidad del software libre dijo «por favor, enviadnos las respuestas también a nosotros». Así que publicaron esas respuestas, que generalmente estaban en contra. Nunca se hubiera supuesto eso a partir el informe que publicó la oficina de patentes del Reino Unido.

Usan un término que ellos llaman «efecto técnico». Este es un término que puede estirarse enormemente. Supuestamente tienes que pensar que significa que una idea sobre un programa sólo sería patentable si está relacionada con actos físicos específicos. Si esa es la interpretación, esto en su mayor parte resolvería el problema. Si las únicas ideas de software que pudieran patentarse fueran aquellas que de verdad estuvieran relacionadas con un resultado técnico o físico particular, que hubieras patentado si no hubieras usado el programa, eso estaría bien. El problema es que puedes estirar ese término. Puedes describir el resultado que obtienes por utilizar cualquier programa como un resultado físico. ¿Cómo se diferencia este resultado físico de cualquier otro? Bueno, se diferencia como resultado de este cómputo. El resultado es que la oficina de patentes del Reino Unido propone algo que parece llevar a que resuelva el problema en su mayor parte, pero en realidad da carta blanca para patentar casi cualquier cosa.

El personal del mismo ministerio también está implicada en asuntos referidos al copyright, que realmente no tienen nada que ver con las patentes del software excepto en que están siendo manejadas por la misma gente. (A lo mejor se han visto llevados por el término «propiedad intelectual» para confundir las dos cuestiones.) Se trata de interpretar la reciente directiva de copyright de la UE, una ley horrible como la Digital Millenium Copyright Act en los EE.UU., pero con algo de flexibilidad para que los países decidan cómo se implementa. Reino Unido propone la manera más draconiana posible de implementar esta directiva. Se podría reducir mucho el daño implementándola apropiadamente. Reino Unido quiere maximizar el efecto tiránico de esta directiva. Parece que hay cierto grupo —¿el Departamento de Comercio e Industria?— que debe ser frenado. Es necesario revisar sus actividades y detener la gestación de nuevas formas de poder.

Las patentes de software subordinan a todo desarrollador de software y a todo usuario a una nueva forma de burocracia. Si los negocios que usan ordenadores se

dieran cuenta de cuántos problemas les puede causar esto, se levantarían en armas, y estoy seguro de que podrían detenerlo. A los negocios no les gusta estar subordinados a la burocracia. Hay algunas áreas en las que nos gustaría que el gobierno del Reino Unido hiciera un trabajo más cuidadoso al subordinar ciertos negocios a la burocracia, como en todo lo que se refiere al desplazamiento de animales. Pero en los casos en los que no sirve a otro propósito que a crear monopolios artificiales, de modo que alguien pueda interferir en la creación de software —exprimiendo dinero de los desarrolladores y los usuarios—, deberíamos rechazarlo. Tenemos que hacer conscientes a las empresas de lo que las patentes de software les pueden hacer, y conseguir su apoyo para luchar contra las patentes de software en Europa.

La batalla no ha terminado. Todavía puede ser ganada.

Parte III

Libertad, sociedad y software

Capítulo 17
¿Puedes confiar en tu ordenador?

¿De quién debería recibir órdenes tu ordenador? Mucha gente piensa que sus ordenadores deberían obedecerles a ellos y no a otras personas. Mediante un plan al que llaman «informática de confianza», las grandes corporaciones de los medios de comunicación —incluyendo las compañías cinematográficas y de la industria discográfica— al lado de compañías del ámbito de la informática tales como Microsoft e Intel, planean hacer que su ordenador los obedezca a ellos en lugar de a ti. Los programas propietarios han incluido características malévolas en el pasado, pero este plan las universalizaría.

Software propietario significa, fundamentalmente, que tú no controlas lo que haces; no puedes estudiar el código fuente ni modificarlo. No es sorprendente que hábiles hombres de negocios encuentren medios de control para situarte en una situación de desventaja. Microsoft ha hecho esto varias veces; una versión de Windows fue diseñada para informar a Microsoft sobre todo el software contenido en su disco duro; una reciente actualización de «seguridad» del Windows Media Player requería que los usuarios aceptaran nuevas restricciones. Pero Microsoft no está solo en esto: el software para intercambio de música KaZaa está diseñado de forma que un asociado de negocios de KaZaa pueda alquilar el uso de tu ordenador a sus clientes. Estas características malévolas son normalmente secretas, pero una vez que te enteras de ellas es difícil eliminarlas dado que no dispones del código fuente.

En el pasado, estos fueron incidentes aislados. La «informática de confianza» los haría omnipresentes. «Informática traicionera» sería un nombre más apropiado, en la medida en que el proyecto está diseñado para asegurarse de que sistemáticamente tu ordenador te va a desobedecer. De hecho, está diseñado para que tu ordenador deje de funcionar como un ordenador de propósito general. Cada operación puede requerir de una autorización explícita.

La idea técnica subyacente a la informática traicionara es que el ordenador incluya un dispositivo de encriptación y de firma digital, cuyas claves permanecerían en secreto. (La versión de Microsoft se llama «palladium».) Los programas propietarios usan este dispositivo para controlar qué otros programas puedes ejecutar, a qué documentos o datos puedes acceder y a qué programas se los puedes transferir. Estos programas descargarán continuamente nuevas reglas de autorización a través de Internet, e impondrán dichas reglas automáticamente en tu trabajo. Si no permites que tu ordenador obtenga las nuevas reglas periódicamente de Internet, algunas capacidades dejarán de

funcionar automáticamente.

Por supuesto, Hollywood y las empresas discográficas planean usar la informática traicionera para «DRM» (*Digital Restriction Management*, «Administración de Restricciones Digitales»), de modo que los vídeos y la música descargados puedan ser reproducidos sólo en un ordenador específico. Compartir será completamente imposible, al menos usando los archivos autorizados que tendrás que obtener de dichas compañías. Tú, el público, deberías tener la libertad y la capacidad de compartir esas cosas. (Espero que alguien encuentre la forma de producir versiones no cifradas, de subirlas y compartirlas, así DRM no tendrá un éxito completo, aunque esto no es una excusa para el sistema.)

Hacer imposible el hecho compartir es de por sí bastante malo, pero el asunto es peor. Existen planes para usar el mismo mecanismo para enviar documentos por correo electrónico —produciendo mensajes que desaparecen en dos semanas, o documentos que sólo pueden ser leídos en los ordenadores de determinada empresa.

Imagina que recibes un mensaje de correo electrónico de tu jefe diciéndote que hagas algo que piensas que es arriesgado; un mes después, cuando el tiro sale por la culata no puedes usar el mensaje para demostrar que la decisión no fue tuya. «Ponerlo por escrito» no te protege si la orden está escrita en tinta que desaparece.

Imagina que recibes un mensaje de correo electrónico de tu jefe estableciendo una política que es ilegal o moralmente ultrajosa, como destrozar los documentos de la auditoría de tu empresa, o permitir que una amenaza peligrosa para tu país avance sin control. Ahora, puedes enviar esta noticia a una periodista y exponer esa actividad. Con la informática traicionera, la periodista no será capaz de leer el documento; su ordenador se negará a obedecerla. La informática traicionera se transforma en un paraíso para la corrupción.

Los procesadores de texto como Microsoft Word podrían usar la informática traicionera cuando guardes tus documentos para asegurarte de que ningún procesador de texto de la competencia pueda leerlos. En la actualidad debemos averiguar los secretos del formato de Word mediante laboriosos experimentos, para que los procesadores libres puedan leer documentos de Word. Si Word cifra los documentos usando la informática traicionera cuando los guarda, la comunidad del software libre no tendrá la posibilidad de desarrollar software para leerlos —y si pudiéramos, esos programas podrían ser prohibidos por la *Digital Millennium Copyright Act*.

Los programas que usen la informática traicionera descargarán continuamente nuevas reglas de autorización desde Internet e impondrán dichas reglas en tu trabajo. Si a Microsoft, o al gobierno de los EE.UU., no les agrada lo que dices en un documento, podrán publicar nuevas restricciones instruyendo a todos los ordenadores para que prohíban que alguien lea dicho documento. Cada ordenador obedecerá cuando descargue las nuevas instrucciones. Su escrito estará sujeto a un supresión retroactiva estilo 1984. Hasta usted podría ser incapaz de leerlo.

Podrías pensar que puedes averiguar qué cosas sucias hace una aplicación de la informática traicionera, estudiar cuán dañinas son y decidir si aceptarlas o no. Sería ingenuo aceptarlas, pero el problema es de tal magnitud que no podrías resistir mucho

tiempo. Una vez dependas del uso del programa, estarás enganchado, ellos lo saben; entonces pueden cambiar las condiciones del acuerdo. Algunas aplicaciones descargarán automáticamente actualizaciones que harán algo diferente —y no te darán la posibilidad de elegir si desead la actualización o no.

Hoy por hoy, puedes evitar las limitaciones del software propietario no usándolo. Si ejecutas GNU/Linux u otro sistema operativo libre y si evitas instalar aplicaciones propietarias sobre él, entonces estarás al mando de lo que tu ordenador hace. Si un programa libre tiene una característica malévola, otros desarrolladores de la comunidad la suprimirán y podrán usar la versión corregida. Puedes también ejecutar aplicaciones y herramientas libres en sistemas operativos no libres; esto no te proporciona una plena libertad, pero muchos usuarios lo hacen.

La informática traicionera pone en peligro la existencia de sistemas operativos y aplicaciones libres, en la medida en que ya no podrás ejecutarlas en absoluto. Algunas versiones de la informática traicionera requerirán que el sistema operativo está específicamente autorizado por alguna empresa particular. Los sistemas operativos libres no podrán ser instalados. Algunas versiones de la informática traicionera requerirán que cada programa sea específicamente autorizado por el desarrollador del sistema operativo. No podrás ejecutar aplicaciones libres en tales sistemas. Si averiguas cómo hacerlo y se lo dices a alguien podría constituir un delito.

Existen proyectos de ley en EEUU que requieren que todas los ordenadores soporten informática traicionera y que se prohíba la conexión de ordenadores antiguos a Internet. La CBDTPA (la llamamos *Consume But Don't Try Programming Act*, «Consuma Pero No Trate de Programar») es uno de ellos. Pero incluso si no te obligan legalmente a migrar hacia la informática traicionera, la presión para aceptarla puede ser enorme. Ahora, las personas utilizan por lo general el formato Word para comunicarse, aunque esto causa varias clases de problemas[1]. Si sólo una máquina con informática traicionera puede leer los últimos documentos de Word, mucha gente migrará hacia ella, en la medida en que consideren la situación únicamente en términos de acción individual —o lo tomas o lo dejas. Para oponernos a la informática traicionera, debemos unirnos y confrontar la situación como una elección colectiva.

Para mayor información sobre la informática traicionera véase `http://www.cl.cam.ac.uk/users/rja14/tcpa-faq.html`.

Bloquear la informática traicionera requerirá que se organicen un gran número de ciudadanos. ¡Necesitamos tu ayuda! La Electronic Frontier Foundation y Public Knowledge están organizando campañas contra la informática traicionera, así como también el Digital Speech Project esponsorizado por la FSF. Por favor, visita estos sitios Web para poder sumarte y apoyar su labor. También puedes ayudar escribiendo a las oficinas de asuntos públicos de Intel, IBM, HP/Compaq, o cualquiera a quien le hayas comprado un ordenador, explicándole que no quieres ser presionado a comprar sistemas de informática «fiable», con lo que no estás de acuerdo en que ellos los produzcan. Puedes ejercer la presión del poder del consumidor. Si haces esto, por favor envía copias de tus cartas a las organizaciones antes citadas.

[1] Véase `http://www.gnu.org/no-word-attachments.html`

Postscriptum

El proyecto GNU distribuye GNU Privacy Guard, un programa que implementa cifrado de clave pública y firmas digitales, que puede utilizase para enviar mensajes de correo electrónico seguros y privados. Es muy ilustrativo examinar cómo GPG se diferencia de la informática traicionera y ver qué hace a una tan útil y a la otra tan peligrosa.

Cuando alguien usa GPG para enviarte un documento cifrado y usas GPG para decodificarlo, el resultado es un documento no cifrado que puedes leer, reenviar, copiar e incluso re-cifrar para enviarlo de forma segura a un tercero. Una aplicación de informática traicionera te dejaría leer las palabras en la pantalla, pero no producir un documento no cifrado que pudiera usarse de otra forma. GPG, un paquete de software libre, pone las funciones de seguridad a disposición de los usuarios: *los usuarios* usan *el programa*. La informática traicionera está diseñada para imponer restricciones a los usuarios: es *ella* la que usa a los *usuarios*.

Capítulo 18

Por qué el software debe ser libre[1]

Introducción

La existencia del software plantea inevitablemente la pregunta de qué decisiones deberían tomarse respecto a su uso. Por ejemplo, supongamos que una persona que tiene una copia de un programa, se encuentra con otra que desearía tener otra copia del mismo. Es posible copiar el programa; ¿quién debería decidir si esto se lleva a cabo o no? ¿Las personas involucradas? ¿O un tercero, llamado «propietario»?

Por lo general, los desarrolladores de software consideran estos problemas basándose en que el criterio para responder a esta pregunta es el de maximizar los beneficios del desarrollador. El poder político del sector empresarial ha llevado al gobierno a adoptar igualmente este criterio y esta respuesta que proponen los desarrolladores: que el programa tiene un dueño, generalmente una compañía asociada a su desarrollo.

Me gustaría considerar el mismo problema usando un criterio diferente: la prosperidad y la libertad del público en general.

La respuesta no puede provenir de la ley vigente —la ley debería ajustarse a la ética y no al revés. Tampoco el día a día resuelve el problema, a pesar de que puede sugerir algunas soluciones posibles. La única forma de juzgar es observar quién se ve beneficiado y quién se ve perjudicado mediante el reconocimiento de los propietarios de software, por qué y en qué medida. En otras palabras, deberíamos realizar un análisis del tipo coste-beneficio en nombre de la sociedad como un todo, teniendo en cuenta la libertad individual así como la producción de bienes materiales.

En este ensayo, describiré los efectos provocados por el reconocimiento de los propietarios y mostrará que los resultados son perjudiciales. Mi conclusión es que los programadores debemos dedicarnos a animar a otros a compartir, redistribuir, estudiar y mejorar el software que escribimos, en otras palabras, animar a escribir *software libre*.[2]

[1] Escrito originalmente en 1992

[2] El adjetivo *libre* en «software libre» hace referencia a la libertad, no al precio; el precio pagado por una copia de un programa libre puede ser cero, bajo o —en muy pocas ocasiones— bastante alto.

Cómo los propietarios justifican su poder

Aquellos que se benefician del sistema actual, en el que los programas son concebidos como propiedad privada, esgrimen dos argumentos en favor de su derecho de ser propietarios de programas: el argumento emocional y el argumento económico.

El argumento emocional es del tipo: «Pongo mi sudor, mi corazón, mi alma en este programa. ¡Proviene de *mí*, es *mío*!»

Este argumento no requiere una refutación seria. El sentimiento de apego puede ser cultivado por los programadores cuando les convenga, pero no es inevitable. Considérese, por ejemplo, cuán deseosos firman y ceden sus derechos sobre el programa a una gran empresa a cambio de un salario; misteriosamente el apego emocional se desvanece. Por el contrario, considérense a los grandes artistas y artesanos de la época medieval, que ni siquiera firmaban sus trabajos. Para ellos, el nombre del artista no era importante. Lo que importaba era que el trabajo se había hecho —y el propósito al que servía. Esta visión ha prevalecido durante cientos de años.

El argumento económico es del tipo: «Quiero ser rico —normalmente expresado de manera poco precisa como "tengo que vivir de algo"— y si no me dejas enriquecerme programando, entonces no programaré. Todo el mundo es como yo, de manera que nadie programará jamás. ¡Y te encontrarás con que no tienes programas!» Esta amenaza suele venir disfrazada como un amigable y sabio consejo.

Explicaré más tarde por qué esta amenaza es algo completamente absurdo. Antes me gustaría presentar un presupuesto implícito que está mucho más presente en otra formulación del mismo argumento.

Esta formulación empieza comparando la utilidad social del software propietario con la utilidad que se derivaría de no tener software y entonces llega a la conclusión de que el software propietario es, en general, beneficioso y que debería ser promovido. La falacia reside aquí en comparar solamente dos posibilidades —software propietario *versus* ausencia de software— y suponer que no existen otras posibilidades.

En un sistema en el que impera la propiedad intelectual, el desarrollo del software se encuentra generalmente vinculado a la existencia de un dueño que controla el uso de ese software. Mientras exista este vínculo, nos enfrentamos continuamente a la elección entre software propietario o nada. Sin embargo, esta vínculo no es inherente ni tampoco inevitable; es más bien consecuencia de una decisión política sociolegal específica que aquí estamos cuestionando: la decisión de que el software tenga propietarios. Formular la elección entre software propietario y ausencia de software implica empobrecer la cuestión.

El argumento en contra de la propiedad del software

La pregunta que se nos plantea es: «¿debería el desarrollo del software estar vinculado a la existencia de propietarios que restrinjan su uso?»

Para resolver este problema, tenemos que evaluar el efecto en la sociedad de cada una las dos opciones *independientemente*: el efecto de desarrollar software —sin tener en

cuenta la manera en que se redistribuye— y el efecto de restringir su uso —suponiendo que el software ha sido desarrollado. Si una de estas actividades es beneficiosa y la otra es perjudicial, deberíamos deshacernos de esta doble actividad y utilizar sólo la beneficiosa.

En otras palabras, si restringir la distribución de un programa ya desarrollado es perjudicial para la sociedad en su conjunto, entonces un desarrollador de software con una orientación ética debería rechazar esta opción.

Para determinar el efecto de restringir el derecho a compartir, necesitamos comparar los beneficios para la sociedad de un programa restringido —propietario— con los que ofrece ese mismo programa accesible a todo el mundo. Esto significa comparar dos mundos posibles.

Este análisis también tiene en cuenta el contra-argumento, a veces defendido, de que «los beneficios que se proporcionan al vecino al darle una copia de un programa se cancelan por el perjuicio provocado al propietario». Este contra-argumento presupone que el perjuicio y el beneficio son iguales en magnitud. El análisis implica la comparación de ambas magnitudes y muestra que el beneficio es mucho mayor que el perjuicio.

Para clarificar todo esto, vamos a aplicarlo a otro ámbito: la construcción de carreteras.

Pudiera ser que la financiación para construir todas las carreteras proviniese de los peajes. En consecuencia, nos encontraríamos puntos de peaje en cada esquina. Un sistema de este tipo generaría incentivos a la hora de mejorar las carreteras. También tendría la virtud de obligar a los usuarios de una determinada carretera a que pagasen por ella. Sin embargo, un punto de peaje es un obstáculo artificial para una circulación fluida —artificial, porque no es una consecuencia derivada del funcionamiento de los coches o de las carreteras.

Si comparamos la utilidad de las carreteras libres y de la carreteras con peaje, encontramos que —siendo iguales en todo—, las carreteras sin puntos de peaje son más baratas de construir, más baratas de administrar y más eficientes.[3] En un país pobre, el peaje podría provocar que algunas carreteras fuesen inaccesibles a muchos ciudadanos. De manera que las carreteras sin peajes ofrecen mayores beneficios a la sociedad y un coste menor; por lo tanto son preferibles para la sociedad. De este modo, la sociedad debería elegir financiar las carreteras de otra forma, y no mediante peajes. El uso de las carreteras, una vez construidas, debería ser gratuito.

Cuando los defensores de los peajes, los presentan como *meros* recaudadores de fondos, distorsionan la elección que existe de verdad. Los peajes incrementan los fondos públicos, pero hacen algo más: degradan, de hecho, la carretera. La carretera de peaje no es tan buena como la carretera libre; que se nos proporcionen más carreteras o ca-

[3]Los problemas asociados a la contaminación y a la congestión del tráfico no modifican esta conclusión. Si queremos encarecer la conducción, para desanimar la conducción en general, no deberíamos recurrir a las peajes que contribuyen a aumentar la contaminación y la congestión. Un impuesto sobre la gasolina es mucho mejor. Del mismo modo, no es relevante el deseo de aumentar la seguridad en una carretera limitando el máximo de velocidad. Una carretera de libre acceso aumenta la media de velocidad evitando las paradas y los atascos, sea cual sea el límite de velocidad.

rreteras técnicamente superiores puede muy bien no ser una mejora si implica sustituir carreteras libres por carreteras de peaje.

Por supuesto, la construcción de una carretera gratuita cuesta dinero, que de alguna manera la gente debe pagar. Sin embargo, esto no implica la inevitabilidad de los peajes. Nosotros, que en ambos casos pagamos, obtendremos mayores beneficios de nuestro dinero si compramos una carretera gratuita.

No quiero decir que una carretera de peaje sea peor que la ausencia de carreteras. Eso sería verdad si el peaje fuese tan alto que casi nadie pudiese usarla —pero esta es una política improbable para un recaudador de impuestos. Sin embargo, en tanto que los peajes suponen pérdidas de tiempo y molestias considerables, es mejor conseguir el dinero de una manera menos obstructora.

Para aplicar este mismo argumento al desarrollo del software, mostrará ahora que introducir «peajes» en el software le cuesta caro a la sociedad: hace que se encarezca la construcción de los programas, encarece la distribución y los hace menos satisfactorios y eficientes con relación a su uso. De lo que se deduce que la construcción de programas debería promoverse de alguna otra forma. Más tarde, continuará explicando otros métodos de promoción y —hasta donde sea de verdad necesario— financiación del desarrollo de software.

El perjuicio ocasionado por obstaculizar el software

Consideremos por un momento que un programa ha sido desarrollado y que cualesquiera pagos necesarios para su desarrollo han sido realizados; ahora la sociedad debe decidir entre convertirlo en propietario o permitir que se use y comparta libremente. Supóngase que la existencia del programa y su disponibilidad es algo deseable.[4]

Las restricciones sobre la distribución y modificación del programa no pueden facilitar su uso. Sólo pueden interferir en él. Así que el efecto solamente puede ser negativo. ¿Pero cuánto? ¿Y de qué tipo?

Existen tres niveles diferentes de daño material que provienen de esta interferencia:

- Un menor número de personas usa el programa.
- Ninguno de los usuarios puede adaptar o arreglar el programa.
- Otros desarrolladores no pueden aprender del programa, o basar un trabajo nuevo en él.

[4]Podríamos considerar perjudicial un programa determinado y por lo tanto desear que no es disponible en absoluto, como ocurre con la base de datos personales de Lotus Marketplace, que se retiró del mercado gracias a las protestas del público. Buena parte de lo que vengo diciendo no es aplicable a este caso, pero no tiene sentido abogar por la existencia de propietarios por el simple hecho de que el propietario limite la disponibilidad del programa. El propietario no limitará completamente la disponibilidad de un programa, tal y como nos gustaría, en el caso de un programa cuyo uso se considere destructivo.

Cada nivel de perjuicio material lleva asociado un perjuicio psico-social. Me refiero al efecto que tiene las decisiones de la gente sobre sus sentimientos, actitudes y predisposiciones posteriores. Estos cambios en la manera de pensar de la gente tendrán un efecto posterior en sus relaciones con sus conciudadanos y pueden acarrear consecuencias efectivas.

Los tres niveles de perjuicio material desaprovechan parte del valor que el programa podría proporcionar, pero no lo pueden reducir a nada. Si desaprovechan casi todo el valor del programa, entonces el hecho de escribir el programa perjudica a la sociedad en la medida en que se dedicó un esfuerzo en escribir el programa. Se podría decir que aquel programa que produce beneficios al venderse debe proporcionar algún tipo de beneficio material directo.

Sin embargo, teniendo en cuenta el perjuicio psico-social asociado, no existe límite al perjuicio que puede llegar a ocasionar el desarrollo de software propietario.

Obstaculizar el uso de programas

El primer nivel de perjuicio impide el simple uso del programa. Una copia del programa tiene un coste marginal nulo —y se puede pagar este coste realizando esta copia personalmente—, de manera que en un mercado libre tendría un precio casi nulo. El pago por una licencia es un desincentivo significativo a la hora de usar el programa. Si un programa de gran utilidad es propietario, mayor será la cantidad de gente que no lo use.

Es fácil mostrar que la contribución total que un programa proporciona a la sociedad se reduce al asignársele un propietario. Cada usuario potencial del programa, enfrentado al hecho de tener que pagar para usarlo, puede escoger entre pagar o renunciar a usar el programa. Cuando un usuario escoge pagar, esto es en realidad una transferencia nula de riqueza entre las dos partes. Pero cada vez que alguien elige no usar el programa, se provoca un perjuicio a esa persona sin que nadie salga beneficiada. La suma entre números negativos y ceros es siempre negativa.

Pero esto no reduce la cantidad de trabajo que lleva *desarrollar* el programa. Como resultado, la eficiencia del proceso entero, medida en satisfacción del usuario final por hora de trabajo, se reduce.

Esto muestra la diferencia crucial entre las copias de programas y los coches, las sillas o los bocadillos. No existe una copiadora de objetos materiales fuera de la ciencia ficción. Pero los programas son fáciles de copiar; cualquiera puede producir tantas copias como desee, con muy poco esfuerzo. Esto no es cierto para objetos materiales porque la materia se conserva: cada copia nueva tiene que generarse con materia prima de la misma forma en que se construyó la primera copia.

Con objetos materiales, un desincentivo a la hora de usarlos tiene cierto sentido, porque un menor número de objetos comprados implica menos materia prima y menos trabajo para producirlos. Es cierto que generalmente existe un coste inicial, un coste de desarrollo, que se extiende sobre el proceso de producción. Pero mientras el coste marginal de producción puede ser significativo, añadir una participación en el coste de

desarrollo no produce una diferencia cualitativa. Y no requiere restricciones sobre la libertad de los usuarios normales.

Sin embargo, imponer un precio en algo que, de otra manera, podría ser gratuito, es un cambio cualitativo. Un pago impuesto unilateralmente sobre la distribución del software provoca un gran desincentivo.

Más aún, la producción centralizada tal y como se practica en nuestros días es ineficiente incluso en términos de distribución de las copias de software. Este sistema incluye enviar discos o cintas magnéticas en embalajes superfluos, mandar grandes cantidades de ellos a lo largo y ancho del mundo y almacenarlos para venderlos. Este costo se presenta como derivado de hacer negocios; en realidad, es una parte del gasto inútil causado por el hecho de tener dueños.

La cohesión social dañada

Suponga que tanto usted como su vecino consideraran útil la ejecución de un cierto programa. En un pacto ético con su vecino, seguramente entenderíais que una solución apropiada de la situación posibilitaría que los dos usasen el programa. Una propuesta que permitiese usar el programa solo a uno, restringiendo al otro, es discriminatoria; a ninguno de los dos, usted o su vecino, les debería de parecer aceptable.

Firmar una licencia típica de software implica traicionar a tu vecino: «Prometo privar a mi vecino de este programa para que yo pueda tener una sola copia para mí.» Las personas que toman estas decisiones sienten una presión psicológica interna que les empuja a justificarlas degradando la importancia de ayudar al prójimo —de tal forma que el espíritu público sale perjudicado. Se trata de un daño psico-social asociado con el daño material provocado por la desincentivación de usar el programa.

Muchos usuarios admiten inconscientemente que resulta erróneo negarse a compartir, así que deciden ignorar las licencias y las leyes, y comparten el programa de todas formas. Pero a menudo se sienten culpables haciéndolo. Saben que deben infringir las leyes para poder ser buenos vecinos, pero siguen considerando que las leyes tienen autoridad y concluyen que ser un buen vecino —dado que lo son— es algo malo o de lo que sentirse avergonzados. Se trata, también, de un tipo de daño psico-social, pero se puede escapar de ello decidiendo que las licencias y las leyes no tienen fuerza moral alguna.

Los programadores también sufren ese daño psico-social al saber que a muchos usuarios se les impedirá aprovechar su trabajo. Esto conduce a una actitud de cinismo o de autoengaño. Un programador puede describir de manera entusiasta un trabajo que considera técnicamente interesante, y cuando se le pregunta: «¿Se me dejará usar el programa?», se vuelve cabizbajo y admite que la respuesta es no. Para evitar desalentarse, o bien la mayor parte del tiempo ignora este hecho, o adopta una cínica postura diseñada para menoscabar su importancia.

Desde la era Reagan,[5] la principal fuente escasez de los Estados Unidos no es la de las innovaciones técnicas sino más bien la del deseo de trabajar juntos por el bien público. No tiene sentido alentar lo primero a expensas de esto último.

Obstruir la adaptación personalizada de programas

El segundo nivel de perjuicio material es la imposibilidad de adaptar los programas. La posibilidad de modificar el software es una de las grandes ventajas frente a formas mas antiguas de tecnología. Sin embargo, la mayoría del software comercial disponible no lo es en términos de «modificabilidad», ni siquiera después de comprarlo. Puedes decidir tomarlo o dejarlo, como una caja negra —tan solo eso.

El programa que ejecutas consiste en una serie de números cuyo significado permanece oscuro. Nadie, ni siquiera un buen programador, puede cambiar fácilmente esos números para hacer que el programa haga algo diferente.

Los programadores trabajan normalmente con el «código fuente» del programa, que se encuentra escrito en un lenguaje de programación como Fortran o C. Recurren a nombres que designan los datos usados y las partes del programa y representan operaciones con símbolos tales como «+» para la suma y «-» para la resta. Está diseñado para ayudar a los programadores a leer y modificar los programas. He aquí un ejemplo; un programa que calcula la distancia entre dos puntos en un plano:[6]

```
float
distance (p0, p1)
 struct point p0, p1;

{
  float xdist = p1.x - p0.x;
  float ydist = p1.y - p0.y;
  return sqrt (xdist * xdist + ydist * ydist);
}
```

Aquí está ese mismo programa en formato ejecutable[7] en el ordenador que suelo utilizar:

[5]Ronald Reagan, presidente número 40 de los Estados Unidos, es famoso por haber realizado recortes en numerosos programas sociales. También creó un política económica, llamada a menudo *trickle down economics*, considerada por muchos un fracaso.

[6]Comprender cómo funciona este código fuente no es lo importante; lo que es realmente importante es observar que el código fuente esté escrito a un nivel de abstracción que sea claramente comprensible.

[7]Obsérvese la no comprensibilidad del formato ejecutable; dar sentido al formato ejecutable es claramente mucho más complejo que el código fuente de más arriba.

```
1314258944      -232267772      -231844864      1634862
1411907592      -231844736      2159150         1420296208
-234880989      -234879837      -234879966      -232295424
1644167167      -3214848        1090581031      1962942495
572518958       -803143692      1314803317
```

El código fuente es útil —potencialmente al menos— para cualquier usuario de un programa. Pero a la mayoría de los usuarios no se les permite tener copias del código fuente. Generalmente el código fuente de un programa propietario es guardado en secreto por el propietario, por miedo a que cualquier otro pueda aprender algo de él. Los usuarios reciben solamente ficheros de números incomprensibles, que el ordenador se encargará de ejecutar. Esto quiere decir que solo el propietario del programa puede modificar el programa.

Una amiga me habló una vez que trabajó como programadora en un banco durante seis meses, escribiendo un programa similar a otro que se podía obtener comercialmente. Pensaba que si hubiese tenido acceso al código fuente de ese programa comercial lo podría haber adaptado fácilmente a las necesidades del banco. El banco estaba dispuesto a pagar por ello, pero no le estaba permitido hacerlo —el código fuente era secreto. De manera que tuvo que dedicar seis meses de trabajo de desarrollo, un trabajo que aparece contabilizado en el Producto Interior Bruto pero que realmente fue un desperdicio.

El laboratorio de Inteligencia Artificial del MIT (AI lab) recibió de regalo una impresora gráfica de Xerox hacía 1977. Corría con software libre al que añadimos bastantes mejoras útiles. Por ejemplo, el software notificaba inmediatamente al usuario cuando el trabajo de impresión se había realizado. Cuando la impresora tenía un problema, como una obstrucción de papel o falta de papel, el software lo notificaba inmediatamente a todos los usuarios que tuviesen trabajos pendientes. Estas mejoras facilitaban el trabajo.

Más tarde Xerox donó al Laboratorio de IA una impresora nueva, más rápida, una de las primeras impresoras láser. Funcionaba con software propietario que corría en un ordenador independiente dedicado en exclusiva, de manera que no pudimos añadir ninguna de nuestras mejoras favoritas. Pudimos hacer que enviase una notificación cuando se mandaba un trabajo de impresión al ordenador dedicado a la impresora, pero no cuando el trabajo se había impreso —y generalmente el retraso era considerable. No había forma de saber cuando el trabajo se había impreso; lo único que podías hacer era adivinarlo. Y nadie sabía nunca cuando se atascaba el papel, así que a menudo la impresora se quedaba fuera de servicio por espacio de una hora.

Los programadores de sistema del laboratorio del IA Lab estaban capacitados para arreglar aquellos problemas, probablemente tan capacitados como los autores originales del programa. Xerox no mostró interés en arreglar aquellos fallos y prefirió advertirnos de los problemas, de manera que nos vimos forzados a aceptarlos. Nunca se arreglaron.

La mayoría de los programadores buenos han experimentado esta frustración. El banco podía permitirse resolver un problema escribiendo un programa nuevo partiendo de cero, pero un usuario corriente, no importa lo capacitado que esté, sólo puede arrojar la toalla.

Arrojar la toalla provoca un daño psicosocial —al espíritu de independencia. Es desmoralizante vivir en una casa que no puedes arreglar para adecuarla a tus necesidades. Lleva a la resignación y al retraimiento, que pueden extenderse a otros ámbitos de tu vida. La gente que padece de esta manera no se encuentran a gusto y no realiza un buen trabajo.

Imagínese cómo sería si las recetas de cocina se guardasen de la misma manera que el software. Uno se podría preguntar: «¿Cómo cambio esta receta de manera que no tenga sal?». De tal forma que el gran chef respondiese: «¿Cómo se atreve a insultar mi receta, mi creación y mi paladar, manoseándola? ¡No tiene usted el juicio necesario para cambiar mi receta y hacer que salga bien!»

«¡Pero mi doctor me ha prohibido tomar sal! ¿Qué puedo hacer? ¿Va a quitar usted la sal por mí?»

«Me encantaría hacer eso; mis honorarios son de sólo 50.000 dólares». (Las tasas suelen ser grandes debido a la posición de monopolio sobre los modificaciones.) «De todas formas, ahora mismo no tengo tiempo. Estoy ocupado con una comisión para diseñar una nueva receta de galleta marítima para el departamento de Marina. Estaré contigo más o menos en dos años».

Obstaculizar el desarrollo del software

El tercer nivel de daño material afecta al desarrollo del software. El desarrollo del software normalmente era el resultado de un proceso evolutivo, en el que una persona cogía un programa existente y reescribía algunas partes añadir una función nueva, y entonces otra persona reescribía algunas partes más para añadir otra más; en algunos casos, este proceso transcurría durante un periodo de veinte años. Mientras tanto, algunas partes de ese programa eran «canibalizadas» para constituir el comienzo de otros programas.

La existencia de propietarios impide este tipo de evolución, hace necesario empezar desde cero cuando se quiere desarrollar un programa. También impide a los nuevos programadores estudiar los programas disponibles para aprender técnicas útiles o incluso ver cómo están estructurados los programas de mayor envergadura.

Los propietarios también dificultan el aprendizaje. He conocido estudiantes brillantes en ciencia informática que nunca han visto el código fuente de un programa extenso. Puede que fueran buenos escribiendo pequeños programas, pero no pueden empezar a aprender las diferentes habilidades necesarias para escribir programas extensos si no pueden ver cómo lo han hecho otros.

En cualquier campo intelectual, uno puede conseguir metas más elevadas apoyándose en otros. Pero esto ya no se permite por lo general en el campo del software —sólo puedes apoyarte en otros *en tu propia empresa*.

El daño psicosocial asociado afecta al espíritu de cooperación científica, que normalmente era tan intensa que los científicos seguían cooperando incluso cuando sus países entraban en guerra. En este sentido, los oceanógrafos japoneses que abandonaron su laboratorio en una isla del Pacífico preservaron cuidadosamente su trabajo en el momento de la invasión de los marines de los EE.UU. y dejaron una nota pidiendo que lo guardaran bien.

El conflicto por la obtención de beneficio ha destruido lo que se salvó del conflicto internacional. Hoy en día, científicos de numerosas disciplinas no publican lo suficiente en sus trabajos para permitir a otros repetir el experimento. Publican solamente aquello que permita a los lectores maravillarse por lo mucho que saben hacer. Esto es así, desde luego, en la ciencia informática, en donde el código fuente de los programas es generalmente secreto.

No importa cómo se restringe el acto de compartir

He discutido sobre los efectos de impedir a la gente que copie, modifique o desarrolle un programa. No he especificado cómo se lleva a cabo esta obstrucción, puesto que no afecta a la conclusión. Como quiera que se haga, mediante protección anticopia, o copyright, o licencias, o encriptación, o tarjetas ROM, o números de serie en el hardware, si tiene *éxito* impidiendo el uso, el perjuicio está hecho.

Los usuarios consideran algunos de estos métodos más repugnantes que otros. Creo que los métodos más odiados son aquellos que cumplen su objetivo.

El software debería ser libre

He argumentado cómo la propiedad de un programa —el poder de restringir las modificaciones o las copias— es obstructiva. Sus efectos negativos son extensos e importantes. Se sigue pues que en la sociedad no deberían existir propietarios de programas.

Otra manera de comprender esto es reconocer que lo que la sociedad necesita es software libre y el software propietario es un pobre sustituto. Promover el sustituto no es una manera lógica de conseguir lo que necesitamos.

Vaclav Havel nos aconsejó: «Trabajad por algo porque es bueno, no simplemente porque tiene probabilidades de éxito». Un negocio que produce software propietario tiene probabilidades de éxito en sus propios y estrechos términos, pero no es lo que beneficia a la sociedad.

Por qué la gente desarrollará software

Si eliminamos el copyright como forma de animar a la gente a desarrollar software, al principio se desarrollará una menor cantidad de software, pero ese software será más útil. No está claro si la satisfacción total del usuario será inferior; pero si esto es

así, o si queremos aumentarla de todas formas, existen otras maneras de promover el desarrollo, exactamente igual que hay formas alternativas a los peajes para conseguir obtener dinero para las carreteras. Antes de que empiece a hablar sobre cómo hacer esto, primero quiero preguntar que grado de promoción artificial es verdaderamente necesario.

Programar es divertido

Existen algunos tipos de trabajo en los que pocos entrarán si no es por dinero; la construcción de carreteras, por ejemplo. Hay otros campos del estudio y del arte en los que existe escasa probabilidad de enriquecerse, en los que la gente entra por fascinación o por que perciben que son valiosos socialmente. Algunos ejemplos son la lógica matemática, la música clásica y la arqueología; y la organización política entre los trabajadores. La gente compite, de forma triste más que incisiva, por las pocas posiciones remuneradas existentes, ninguna de las cuales financiada de forma generosa. Quizás tengan que pagar por la posibilidad de trabajar en ese campo, si pueden permitírselo.

Un campo así puede transformarse de la noche a la mañana si empieza a ofrecer posibilidades de enriquecimiento. Cuando un trabajador prospera, otros demandan las mismas oportunidades. Pronto todos pedirán grandes sumas de dinero por aquello que antes hacían por placer. En un par de años, todo el mundo relacionado con ese campo se burlará de la idea de que ese trabajo se realice sin grandes sumas de dinero a cambio. Aconsejarán a los planificadores sociales que se aseguren de que estos retornos de capital sean posibles, creando privilegios especiales, poderes y monopolios, alegando que son necesarios para lograrlo.

Esta transformación acaeció en el campo de la programación informática durante la década pasada. Hace quince años[8] uno podía encontrarse con artículos sobre la «adicción a los ordenadores»: los usuarios estaban «conectados» y tenían adicciones que les costaban cien dólares por semana. Parecía aceptable que la gente amase tanto la programación como para acabar con sus matrimonios. Hoy en día, se entiende que nadie programe sin recibir una excelente remuneración a cambio. La gente ha olvidado lo que sabía hace quince años.

Llegado el momento en que quienes trabajan en un campo determinado exigen a cambio altas sumas de dinero, el campo en cuestión ya no necesita regirse por esa pasión voluntariosa. La dinámica del cambio puede efectuarse al revés si la sociedad proporciona el empuje inicial. Si anulamos la posibilidad de enriquecerse enormemente, entonces, después de un tiempo, cuando la gente haya reajustado sus actitudes, volverán una vez más a trabajar en ese campo por el placer de hacerlo.

La respuesta a «¿cómo podemos pagar a los programadores?», resulta más fácil cuando nos damos cuenta de que no es una cuestión de pagarles una fortuna. Es más fácil conseguir los fondos necesarios para ganarse la vida simplemente.

[8] Quince años antes de escribir este artículo transcurría el año 1977.

Financiar el software libre

Las instituciones que pagan a los programadores no tienen que ser necesariamente empresas de software. Otras muchas instituciones ya existentes se pueden encargar de ello.

Los fabricantes de hardware saben que es esencial colaborar en el desarrollo de software incluso aun cuando no puedan controlar el uso de ese software. En 1970, la mayoría del software era libre porque no se había considerado la posibilidad de restringirlo. Hoy en día, su creciente voluntad de unirse en consorcios refleja la consideración de que la propiedad del software no es lo que realmente les importa.

Las universidades dirigen bastantes proyectos de programación. Hoy en día, a menudo venden los resultados, cuando en la década de 1970 no lo hacían. ¿Hay alguna duda de que la universidades desarrollarían software libre si estuviese prohibida la venta de software? Estos proyectos podrían estar respaldados por los mismos contratos y subvenciones gubernamentales que ahora respaldan al desarrollo de software propietario.

Lo normal ahora es que los investigadores universitarios obtengan subvenciones para desarrollar un sistema, desarrollarlo casi hasta el punto de completarlo, denominando a eso un producto «acabado» y luego que las empresas realmente lo terminen y lo conviertan en algo útil. A veces declaran «libre» la versión sin acabar; si son profundamente corruptos entonces consiguen una licencia de exclusividad para la universidad. Esto no es un secreto; se admite abiertamente por todos los involucrados. Sin embargo, si los investigadores no se vieran tentados a hacer estas cosas, seguirían investigando de todas formas.

Los programadores que escriban software libre pueden vivir a base de vender servicios relacionados con el software. He sido contratado para trasladar el *Compilador GNU de C* a un hardware nuevo y para construir interfaces de usuario para *GNU Emacs*. (Ofrezco estas mejoras al público una vez acabadas.) También doy clases por las que me pagan.

No soy el único que trabaja de esta manera. Existe una corporación que está creciendo de forma exitosa y se dedica a este tipo de trabajo. Otras empresas proporcionan soporte comercial para el software libre del sistema GNU. Este es el comienzo de una industria independiente de soporte de software —una industria que podría crecer bastante si el software libre se llega a imponer. Proporciona a los usuarios una opción generalmente inaccesible a través del software propietario, excepto a los más ricos.

Nuevas instituciones[9] como la *Free Software Foundation* pueden también subvencionar a los programadores. La mayoría de los fondos de la Fundación provienen de los usuarios que compran disquetes o cintas por correo. El software en disquetes es libre, lo que quiere decir que cualquier usuario tiene la libertad de copiarlo y cambiarlo, pero muchos a pesar de ello pagan por conseguir copias. (Recuérdese que «software libre» se refiere a la libertad, no al precio.) Algunos usuarios encargan cintas magnéticas de las

[9] Recordemos que este artículo fue originalmente escrito en 1992

que ya tienen una copia como una forma de contribución que piensan que merecemos. La Fundación también recibe importantes donaciones de fabricantes de ordenadores.

La *Free Software Foundation* es una sociedad sin ánimo de lucro y sus ingresos se invierten en contratar a tantos programadores como se pueda. Si se hubiese planteado como una empresa, distribuir software libre al público por el mismo precio, proporcionaría ahora una buen estándar de vida a su fundador.

Precisamente porque la Fundación es una sociedad sin ánimo de lucro, los programadores trabajan por la mitad de lo que cobrarían en cualquier otro sitio. Hacen esto porque estamos libres de burocracia y porque encuentran satisfacción sabiendo que su trabajo no encontrará obstáculos a su uso. Y lo que es más importante, lo hacen porque sienten que programar es divertido. Además, los voluntarios han escrito muchos programas útiles para nosotros. (Incluso, han empezado a colaborar escritores técnicos.)

Esto confirma que la programación se encuentra entre los campos más fascinantes, junto con la música y el arte. No debemos temer que nadie quiera programar.

¿Qué deben los usuarios a los desarrolladores?

Los usuarios de software tienen una buena razón para sentirse moralmente obligados a contribuir a su soporte. Los desarrolladores de software libre contribuyen a las actividades de los usuarios, y a largo plazo es justo, a la vez que beneficioso para los usuarios, proporcionar fondos para que esto continúe.

Sin embargo, esto no debería de aplicarse a los desarrolladores de software propietario, ya que el obstruccionismo se merece un castigo más que una recompensa.

De manera que tenemos una paradoja: el desarrollador de software útil tiene el derecho a recibir el apoyo de los usuarios, pero cualquier intento que convierta esta obligación moral en una petición destruye la base de la obligación. Un desarrollador puede o bien merecer una recompensa o pedirla, pero no las dos cosas a la vez.

Creo que un desarrollador con perspectiva ética enfrentado con esta paradoja debe actuar de modo que merezca la recompensa, pero debería asimismo animar a los usuarios a que realicen donaciones. Puede que los usuarios aprendan así a ayudar a los desarrolladores sin coacción, como han aprendido a ayudar a las emisoras de radio o a las cadenas de televisión públicas.

¿Qué es la productividad del software?

Si el software fuese libre seguiría habiendo programadores, pero quizá menos. ¿Sería esto perjudicial para la sociedad?

No necesariamente. Hoy en día las naciones desarrolladas tienen menos granjeros que en 1900, pero no creemos que esto sea malo para la sociedad porque esos agricultores distribuyen más comida a los consumidores que antes. Llamamos a esto mejora de la productividad. El software libre requeriría bastantes menos programadores para satisfacer la demanda, debido al aumento en la productividad del software en todos los niveles:

- El uso más extendido de cada programa que se desarrolla.
- La posibilidad de adaptar programas existentes a configuraciones especiales en lugar de tener que crear los programas *desde cero*.
- Mejor educación de los programadores.
- La eliminación de la duplicación de esfuerzos en el desarrollo.

Aquellos que se oponen a la cooperación, quejándose de que podría producir una reducción en el empleo de los programadores, están, en realidad, oponiéndose al aumento de productividad. Y además estas personas aceptan generalmente la creencia universal de que la industria del software necesita un incremento de su productividad. ¿Cómo es esto posible?[10]

«La productividad del software» puede significar dos cosas diferentes: la productividad general de todo el desarrollo del software o la productividad de proyectos individuales. La productividad general es lo que a la sociedad le gustaría mejorar y la forma más directa de lograrlo es eliminar los obstáculos artificiales a la cooperación, que la reducen. Pero los investigadores que estudian el campo de la «productividad del software» se centran sólo en el segundo y más limitado sentido del término, en donde la mejora precisa de complejos avances tecnológicos.

¿Es inevitable la competencia?

¿Es inevitable que la gente trate de competir y superar a sus rivales en la sociedad? Puede que así sea. Pero la competencia en sí misma no es dañina; lo dañino es el *combate*.

Existen muchas formas de competir. La competencia puede consistir en tratar de conseguir siempre más, en mejorar lo que otros han hecho. Por ejemplo, en el pasado, existía competencia entre los gurús de la programación —competencia que consistía en quién era capaz de producir el ordenador que realizase las cosas más fascinantes o quién era capaz de escribir el programa más corto o más rápido para una determinada tarea. Este tipo de competencia puede beneficiar a todos, *mientras* el espíritu de deportividad se mantenga.

Una competencia constructiva es suficiente para motivar a la gente a realizar grandes esfuerzos. Hay personas que compiten por ver quién es el primero en visitar todos

[10] De acuerdo con Eric Raymond el 95 por ciento de los empleos en la industria del software implica la producción de software de aplicaciones personalizadas, en absoluto destinado a la publicación. Se sigue que incluso si asumimos el peor presupuesto teórico, que no habrá empleo en el desarrollo del software libre —y ahora sabemos ya que algo hay—, el cambio al software libre sólo puede tener un pequeño efecto en el número total de empleos. Existe un gran nicho para la gente que tenga empleo escribiendo software de aplicaciones personalizadas y desarrolle software libre en su tiempo libre. No existe manera de saber si la plena conversión al software libre incrementaría o haría decrecer el número de empleos en el campo del software.

los países de la Tierra; algunos llegan a gastar una fortuna intentándolo. Pero no sobornan a los capitanes de barcos para que dejen desamparados a sus rivales en islas desiertas. No tienen ningún problema en dejar que gane al mejor.

La competencia se convierte en combate cuando los competidores intentan obstaculizarse los unos a los otros en lugar de avanzar por sí mismos —cuando «que gane el mejor» se convierte en «déjame ganar, sea el mejor o no». El software propietario es perjudicial, no porque sea una forma de competición, sino porque es una forma de combate entre los ciudadanos de nuestra sociedad.

La competición en los negocios no es necesariamente un combate. Por ejemplo, cuando dos supermercados compiten, todo su esfuerzo se emplea en mejorar sus actividades, no en sabotear al rival. Pero esto no demuestra un especial compromiso con una ética empresarial; por el contrario, existe un pequeño margen de libertad en esta rama de los negocios carente de violencia física. No todas las áreas de negocio comparten esta misma característica. Preservar información que podría ayudar al avance de todos es una forma de combate.

La ideología empresarial no prepara a la gente para resistir la tentación de combatir a la competencia. Algunas formas de combate han sido prohibidas con leyes antimonopolio, leyes sobre honestidad en publicidad y otras más, pero lejos de generalizarse mediante una repulsa, por principio, hacia el combate en general, los ejecutivos inventan otras formas de combate que no están específicamente prohibidas. Los recursos de la sociedad se despilfarran en el equivalente económico de una guerra civil.

«¿Por qué no nos vamos a Rusia?»

En los Estados Unidos, cualquier partidario de otra cosa que no sea la forma más extrema de *laissez-faire* ha oído a menudo esta acusación. Por ejemplo, es esgrimida contra los defensores de un sistema de sanidad pública, como los que existen en todas las demás naciones industrializadas del mundo libre. Es esgrimida contra los que desean subvenciones al mundo de las artes, también universal en las naciones avanzadas. La idea de que los ciudadanos tienen una obligación con el bien común se identifica en Estados Unidos con el comunismo. ¿Pero son semejantes estas ideas?

El comunismo, tal y como se practicó en la Unión Soviética, era un sistema de control central en donde toda la actividad era dirigida supuestamente por el bien común, pero en realidad en beneficio de los miembros del partido comunista. Y donde los equipos de copia estaban estrechamente vigilados para prevenir posibles copias ilegales.

El sistema de copyright sobre el software de Estados Unidos ejerce un control central sobre la distribución de un programa y protege los equipos de copia con sistemas automatizados de protección anticopia, de forma que pueda evitarse la copia ilegal.

Por el contrario, yo trabajo para construir un sistema donde la gente sea libre para decidir sus propias acciones; en particular, libre para ayudar a sus vecinos y libre para alterar y mejorar las herramientas con las que trabajan en su vida cotidiana. Un sistema basado en la cooperación voluntaria y en la descentralización.

Así, si fuésemos a juzgar posturas por su parecido al comunismo ruso, son los propietarios del software quienes son comunistas.

La cuestión de las premisas

En este texto, parto del supuesto de que un usuario de software no es menos importante que un autor, o incluso que el jefe del autor. En otras palabras, sus intereses y necesidades tienen igual peso cuando se trata de dilucidar qué decisión es mejor.

Esta premisa no es aceptada universalmente. Muchos sostienen que la persona que contrata al autor es fundamentalmente más importante que ningún otro. Dicen, por ejemplo, que el propósito de que existan propietarios de software es dar al que contrata al autor la ventaja que se merece —independientemente de como puede afectar esto al público.

No tiene sentido tratar de demostrar o invalidar estas premisas. La prueba necesita premisas compartidas. Así que la mayoría de lo que digo está destinado sólo a aquellos que comparten mis premisas o que al menos están interesados en cuáles son sus consecuencias. Para aquellos que crean que los propietarios son más importantes que nadie, este documento es simplemente irrelevante.

Pero, ¿por qué aceptaría un gran número de estadounidenses una premisa que eleva en importancia a algunas personas sobre el resto del mundo? En parte debido a la creencia de que esta premisa forma parte de las tradiciones legales de la sociedad estadounidense. Algunas personas sienten que poner en duda esta premisa implica cuestionar los fundamentos de la sociedad.

Es importante ser consciente de que esta premisa no forma parte de nuestra tradición legal. Nunca lo fue.

Así, la Constitución dice que el propósito del copyright es «promover el progreso de la ciencia y de las artes útiles». El Tribunal Supremo ha discutido sobre esto, dictando en el caso «Fox Film contra Doyal» que «el único interés del los Estados Unidos y el objetivo principal por el que se otorga el monopolio [del copyright] descansa en los beneficios generales obtenidos por el público gracias al trabajo de los autores».

No estamos obligados a estar de acuerdo con la Constitución o con el Tribunal Supremo. (En un momento dado, los dos perdonaron el esclavismo.) De este modo, sus posiciones no rechazan la premisa de la supremacía del propietario. Pero espero que, la conciencia de que esta suposición es radicalmente conservadora, más que tradicional, debilite su poder.

Conclusión

Nos gusta pensar que nuestra sociedad promueve la buena vecindad, pero cada vez que recompensamos a alguien por su obstruccionismo o admiramos a otro por haberse enriquecido por esta vía, enviamos la señal opuesta.

La acumulación de software es una expresión de nuestra predisposición general a la indiferencia con respecto al bienestar de la sociedad y a favor del bien personal.

Podemos observar esta indiferencia, desde Ronald Reagan a Jim Bakker,[11] desde Ivan Boesky[12] a Exxon,[13] desde la falta de bancos a la de colegios. Podemos medirla por el número de personas sin hogar y la gente encarcelada. El espíritu antisocial se nutre de sí mismo, porque cada vez que comprobamos que la gente no nos ayudará, más fútil nos parece ayudarlos a ellos. Y así la sociedad degenera en una jungla.

Si no queremos vivir en una jungla, debemos cambiar nuestras formas de comportarnos. Debemos empezar enviando el mensaje de que un buen ciudadano es aquel que colabora cuando es apropiado, no aquel que logra éxito cuando roba a los demás. Espero que el movimiento por el software libre pueda contribuir a esto: al menos en un área, reemplazaremos la jungla por un sistema más eficiente que anime y se base en la cooperación voluntaria.

[11]Jim Bakker recaudó millones de dólares de la televisión para sus grupos religiosos Heritige USA, PTL y The Inspirational Network en la década de 1980. Fue encarcelado por fraude en la financiación de PTL y sentenciado a 45 años de cárcel.

[12]Ivan Boesky fue enviado a prisión por tráfico en la década de 1980 y multado con 100 millones de dólares. Es famoso por haber dicho en una ocasión: «La avaricia es buena. Quiero que sepáis que pienso que la avaricia es saludable. Puedes ser avaro y todavía sentirte bien contigo mismo».

[13]En 1980 el Exxon Valdez causó el mayor derrame de petróleo en el mundo sobre la costa de Alaska, provocando un daño inconmensurable. La limpieza y las indemnizaciones les han costado más de 100.000 millones de dólares hasta la fecha.

Capítulo 19

Copyright y globalización en la era de las redes informáticas[1]

DAVID THORNBURN (MODERADOR). Nuestro conferenciante de hoy, Richard Stallman,[2] es una figura legendaria en el mundo de la informática, y mi experiencia al tratar de encontrar una persona que comparta el estrado con él fue instructiva. Un distinguido profesor del MIT me dijo que Stallman debe ser entendido como un personaje carismático de una parábola bíblica —una especie de anécdota o lección del Antiguo Testamento. «Imagínate —me dijo— un Moisés o un Jeremías... mejor un Jeremías». Y yo dije «bien, eso es muy admirable. Suena maravilloso. Confirma mi idea acerca del tipo de contribución que ha hecho al mundo. Entonces, ¿por qué no quieres compartir el estrado con él?». Su respuesta: «como Jeremías o Moisés, él simplemente me apabullaría. No querría aparecer en el mismo programa que él, pero si me pidieras que nombre a cinco personas vivas en el mundo que realmente nos hayan ayudado a todos nosotros, Richard Stallman sería una de ellas».

RICHARD STALLMAN. Debería empezar explicando por qué me negué a permitir que esta conferencia sea transmitida en directo vía Internet, en caso de que no haya sido plenamente aclarada esta cuestión: el software que utilizan para transmitir imagen y sonido en vivo por Internet requiere que el usuario descargue cierto software para recibir la transmisión. Ese software no es software libre. Está disponible a precio cero pero sólo como «ejecutable», que es un misterioso montón de números.

Lo que hace es secreto. No lo puedes estudiar, no lo puedes cambiar y ciertamente no puedes publicarlo en tu propia versión modificada. Y éstas están entre las libertades que son esenciales en la definición de «software libre».

Entonces, si voy a ser un defensor sincero del software libre, difícilmente podría andar dando discursos y ejercer presión sobre la gente para que use software no libre. Estaría socavando mi propia causa. Y si yo no demuestro que me tomo en serio mis principios, no puedo esperar que nadie más los tome en serio.

[1]Lo que sigue es una transcripción corregida de la conferencia dictada en el MIT, en el Communications Forum, el jueves 19 de abril de 2001.

[2]Lo que sigue es una transcripción corregida de la conferencia dictada en el MIT, en el Communications Forum, el jueves 19 de abril de 2001.

Sin embargo, esta charla no es acerca del software libre. Después de haber trabajado en el movimiento del software libre durante muchos años y de que la gente haya comenzado a usar algunas partes del sistema operativo GNU, empecé a ser invitado a dar conferencias, en las que la gente empezó a preguntarme: «Bueno, ¿de qué manera las ideas sobre la libertad para los usuarios de software pueden generalizarse a otro tipo de cosas?». Y, por supuesto, alguna gente hacía preguntas tontas como «¿debería ser libre el hardware?», «¿este micrófono debería ser libre?».

Bien, ¿esto qué significa? ¿Deberías ser libre de copiarlo y modificarlo? Si compras un micrófono, nadie te va a impedir modificarlo. Y copiarlo... nadie tiene un copiador de micrófonos. Fuera de Star Trek, esas cosas no existen. Puede ser que algún día haya analizadores y ensambladores nanotecnológicos, y entonces estas cuestiones de si eres libre o no de hacer copias realmente adquirirán importancia. Veremos empresas agroindustriales intentando impedir que la gente copie alimentos y eso se va a convertir en una cuestión política de primer orden; si es que esa capacidad tecnológica llega a existir. No sé si ocurrirá, de momento es sólo especulación.

Pero para otras formas de información, se puede traer el asunto a colación, puesto que cualquier clase de información que pueda ser almacenada en un ordenador, de forma concebible, puede ser copiada y modificada. Así que las cuestiones éticas del software libre, la cuestión del derecho de un usuario a copiar y modificar software, son las mismas que las relativas a otras formas de información publicada. Yo no estoy hablando de información privada; digamos, información personal, la cual se supone que nunca debería estar disponible para el público. Estoy hablando de los derechos que debieras tener si obtienes copias de cosas publicadas, que no se intenta mantener en secreto.

La historia del copyright

A fin de explicar mis ideas en la materia, quisiera repasar la historia de la difusión de información y del copyright. En el mundo antiguo, los libros se escribían a mano con una pluma y cualquiera que supiera cómo leer y escribir podía copiar un libro casi tan eficientemente como los demás. Cierto que alguien que lo hiciese todo el día probablemente aprendería a hacerlo un poco mejor, pero no había una gran diferencia. Y como las copias se hacían de una en una, no existía una gran economía de escala. Hacer diez copias llevaba diez veces más tiempo que hacer una copia. Tampoco había nada que forzara la centralización; un libro podía copiarse en cualquier lugar.

Ahora bien, debido a las características de esta tecnología, dado que no obligaba a que las copias fueran idénticas, no había en la antigüedad una distinción total entre copiar un libro y escribir un libro. Había cosas en el medio que tenían sentido. Entendían. sí, la idea de autor. Sabían, digamos, que tal obra había sido escrita por Sófocles, pero entre la escritura del libro y su copia había otras cosas útiles que podías hacer. Por ejemplo, podías copiar una parte de un libro, después añadir algunas palabras nuevas, copiar algo más y escribir algo más, y así. Esto se llamaba «escribir un comentario». Era algo muy común y estos comentarios eran apreciados.

Podías también copiar un pasaje de un libro, después añadir algunas palabras y copiar un pasaje de otro libro y añadir más palabras, y así... y esto era hacer un compendio. Los compendios también eran muy útiles. Hay obras que se han perdido, pero algunas de sus partes sobrevivían citadas en otros libros que alcanzaban mayor popularidad que el original. Quizás copiaban las partes más interesantes y así la gente hacía muchas copias de estos fragmentos pero no se molestaban en copiar el original porque no era lo bastante interesante.

Hasta donde yo sé, no había copyright en el mundo antiguo. Cualquiera que quisiera copiar un libro podía copiarlo. Más tarde se inventó la imprenta y los libros empezaron a copiarse en la imprenta. La imprenta no era sólo una mejora cuantitativa en la facilidad de copia, sino que afectaba de manera diferente a los distintos tipos de copiado, ya que introducía una economía de escala inherente. Era mucho trabajo preparar cada página pero mucho menos trabajo hacer varias copias idénticas de la misma. El resultado fue que copiar libros tendió a convertirse en una actividad centralizada y de producción masiva. Las copias de cualquier libro se harían probablemente sólo en unos pocos lugares.

También significó que los lectores ordinarios no podrían copiar libros eficientemente. Sólo si tenías una imprenta podías hacerlo. Se trataba de una actividad industrial.

Durante los primeros siglos de imprenta, los libros impresos no reemplazaron totalmente a los copiados a mano. Las copias artesanales todavía se hacían, a veces por gente rica y a veces por gente pobre. Los ricos lo hacían para tener copias especialmente hermosas, que mostraran lo ricos que eran, y los pobres lo hacían porque quizás no tenían suficiente dinero para comprar una copia impresa, pero tenían tiempo para copiar a mano un libro. Como dice la canción: «el tiempo no es dinero cuando todo lo que tienes es tiempo».

De este modo, el copiado a mano todavía se hacía con una determinada profusión. Creo que fue durante el siglo XIX cuando la impresión se volvió tan barata que incluso la gente pobre podía comprarse libros impresos si sabía leer.

El copyright apareció con el uso de la imprenta y dada la tecnología de la imprenta, tenía el efecto de una regulación industrial. No restringía lo que podían hacer los lectores; restringía lo que podían hacer los editores y los autores. El copyright en Inglaterra inicialmente fue una forma de censura. Tenías que obtener un permiso del gobierno para publicar el libro. Pero la idea cambió. En los tiempos de la Constitución de los Estados Unidos, la gente adoptó una idea diferente del propósito del copyright y creo que esa idea también fue aceptada en Inglaterra.

En la redacción de la Constitución de los EEUU se propuso que a los autores se les debería otorgar un copyright, un monopolio sobre la copia de sus libros. Esta propuesta fue rechazada. En cambio, fue adoptada una propuesta crucialmente diferente: con el fin de promover el progreso, el Congreso podría opcionalmente establecer un sistema de copyright que creara esos monopolios. Estos monopolios, de acuerdo con la Constitución de los EEUU, no existen por el bien de sus propietarios, sino para promover el progreso de la ciencia. Los monopolios se conceden a los autores como un modo de influir en su comportamiento, para lograr que hagan algo que sirva al público.

De este modo, el objetivo es tener más libros escritos y publicados que la gente pueda leer. Y se cree que el copyright contribuye al incremento de la actividad literaria, al incremento de la escritura científica y en otros campos, y que la sociedad aprende así a través de él. Ése es el propósito al que debe servir. La creación de monopolios privados era sólo un medio para procurar un fin y este fin es un fin público.

El copyright en la era de la imprenta era bastante inofensivo, pues se trataba de una regulación industrial. Restringía sólo las actividades de los editores y de los autores. Bueno, en sentido estricto, también los pobres que copiaban libros a mano podrían haber infringido la ley de copyright. Pero nadie trató nunca de forzarlos a respetar el copyright ya que se entendía como una regulación industrial.[3]

El copyright en la era de la imprenta también era fácil de hacer cumplir, dado que tenía que hacerse cumplir sólo donde había un editor y los editores, por su naturaleza, se hacen conocer. Si estás tratando de vender libros, tienes que decirle a la gente dónde ir a comprarlos. No tienes que entrar en la casa de todo el mundo para hacerles respetar el copyright.

En definitiva, el copyright puede haber sido un sistema beneficioso en aquel contexto. El copyright en EEUU es considerado por los especialistas en Derecho como un trato comercial, un contrato entre el público y los autores. El público cede algunos de sus derechos naturales y a cambio se beneficia con la escritura y la publicación de mayor cantidad de libros.

Ahora, ¿es éste un trato ventajoso? Bien, cuando el público en general no puede hacer copias porque sólo pueden hacerse eficientemente en las imprentas —y la mayoría de la gente no tiene imprenta— el resultado es que el público en general está cediendo una libertad que no puede ejercer, una libertad sin ningún valor práctico. Si tienes algo que es un subproducto en tu vida y que es inútil, y tienes la oportunidad de intercambiarlo por algo de algún valor, estás ganando. Así es cómo el copyright pudo haber sido un trato ventajoso para el público en aquella época.

Pero el contexto está transformándose y eso debe cambiar nuestra evaluación ética del copyright. Ahora bien, los principios básicos de la ética no se modifican por los avances de la tecnología; son demasiado fundamentales para estar afectados por tales contingencias. Sin embargo, nuestra decisión sobre cualquier asunto específico depende de las consecuencias de las alternativas disponibles y las consecuencias de una determinada opción pueden cambiar si el contexto cambia. Eso es lo que está ocurriendo con el copyright, la era de la imprenta está llegando a su fin, dando paso gradualmente a la era de las redes informáticas.

Las redes informáticas y la tecnología de la información digital nos están devolviendo a un mundo más parecido a la antigüedad, donde cualquiera que pueda leer y usar la información puede también copiarla y hacer copias casi tan fácilmente como cualquiera. Son copias perfectas y son tan buenas como las que podría hacer cualquiera. De este modo, la centralización y las economías de escala introducidas por la imprenta están desapareciendo.

---

[3]Los estatutos originales hablaban sólo de editar e imprimir. La copia manual estaba totalmente desregulada, muy probablemente porque la regulación estaba dirigida a la industria.

Este contexto cambiante modifica el modo en que funciona la legislación de copyright. Veréis, la ley de copyright ya no actúa como una regulación industrial; ahora es una restricción draconiana sobre el público en general. Solía ser una restricción sobre los editores por el bien de los autores. Ahora, por propósitos prácticos, es una restricción sobre el público en provecho de los editores. El copyright solía ser bastante inofensivo y poco controvertido. No restringía al público en general. Ahora eso ya no es verdad. Si tienes un ordenador, los editores consideran la restricción como su más alta prioridad. El copyright era fácil de hacer cumplir porque era una restricción que pesaba sólo sobre los editores, que eran fáciles de encontrar —y lo que publicaban era fácil de ver. Ahora el copyright es una restricción que pesa sobre cada uno de vosotros. Su cumplimiento requiere vigilancia, intrusión y duros castigos, observamos cómo se está incorporando a la legislación de los EEUU y de otros países.

El copyright solía ser, discutiblemente, un trato ventajoso para el público porque el público estaba cediendo libertades que no podía ejercer. Bueno, ahora sí puede ejercer estas libertades. ¿Qué haces si te has acostumbrado a ceder un subproducto que no te era útil y, de pronto, descubres un uso para él? Puedes, de hecho, consumirlo, usarlo. ¿Qué haces? No lo negocias; te guardas algo. Y eso es lo que el público querría naturalmente hacer. Eso es lo que el público hace cada vez que se le da la oportunidad de expresar su preferencia. Se guarda algo de su libertad y la ejerce. Napster es un gran ejemplo de eso: el público decide ejercer la libertad de copiar en vez de entregarla. Entonces lo que naturalmente debemos hacer para conseguir que la legislación de copyright se ajuste a las circunstancias actuales, es reducir la cantidad de restricciones que pesan sobre el público e incrementar la libertad que el público conserva.

Pero esto no es lo que los editores quieren hacer. Lo que ellos quieren hacer es exactamente lo contrario. Ellos quisieran incrementar los poderes del copyright hasta el punto de que les permita controlar todo el uso de la información. Esto ha llevado a leyes que han concedido un incremento sin precedentes de los poderes de copyright. Se están retirando libertades que el público solía tener en la época de la imprenta.

Por ejemplo, echemos un vistazo a los libros electrónicos. Hay una tremenda cantidad de publicidad sobre los libros electrónicos; difícilmente puedes evitarla. Tomé un vuelo en Brasil y en la revista de a bordo había un artículo diciendo que quizás iba a llevar diez o veinte años hasta que todos nosotros nos pasáramos a los libros electrónicos. Claramente, este tipo de campaña viene de alguien que está pagando por ella. Ahora bien, ¿por qué lo están haciendo? Creo que lo sé. La razón es que los libros electrónicos son la oportunidad de retirar a los lectores de libros impresos algunas de las libertades residuales que tienen y que siempre tuvieron. La libertad, por ejemplo, de prestarle un libro a un amigo, o de tomarlo prestado de una biblioteca pública, o de vender una copia a una librería de viejo, o de comprar una copia anónimamente, sin dejar registrado en una base de datos quién compró ese libro en particular. Y puede que aún el derecho a leerlo dos veces.

Éstas son libertades que los editores quisieran retirar, pero no pueden en el caso de los libros impresos porque sería una usurpación de poder muy obvia y generaría protesta. Entonces encontraron una estrategia indirecta: primero, obtienen la legislación

para retirar esas libertades a los libros electrónicos cuando todavía no hay libros electrónicos, así no hay controversia. No hay usuarios preexistentes de libros electrónicos acostumbrados a sus libertades y dispuestos a defenderlas. Eso es lo que consiguieron con la Digital Millenium Copyright Act en 1998. Entonces introducen los libros electrónicos y gradualmente logran que todos se pasen de los libros impresos a los libros electrónicos, eventualmente el resultado es: los lectores perdieron esas libertades sin que jamás haya habido un instante en el que esas libertades les fueran retiradas y en el que ellos pudieran haber luchado para conservarlas.

Vemos al mismo tiempo esfuerzos para retirarle a la gente la libertad de usar otro tipo de obras publicadas. Por ejemplo, las películas en DVD se publican con un formato cifrado que iba a ser secreto —se suponía que iba a ser secreto— y la única forma en que las compañías cinematográficas iban a darte el formato, de manera que pudieras fabricar un reproductor de DVD, era si firmabas un contrato comprometiéndote a incluir ciertas restricciones en el reproductor, con el resultado de que se impediría al público el ejercicio completo de sus derechos legales. Entonces unos cuantos astutos programadores en Europa encontraron la forma de descifrar los DVD y escribieron un paquete de software libre que podía leer un DVD.[4] Esto hizo posible usar software libre sobre el sistema operativo GNU/Linux para ver el DVD que habías comprado, lo cual es algo perfectamente legítimo. Deberías tener derecho a hacer eso con software libre.

Pero las compañías cinematográficas reclamaron y fueron a juicio. Ya veis, las compañías cinematográficas han hecho un montón de películas en las que hay un científico loco y alguien dice «pero doctor, hay ciertas cosas que se supone que el Ser Humano no debe conocer». Seguramente han visto demasiadas de sus propias películas porque llegaron a creer que el formato de los DVD es algo que el Ser Humano no debía conocer. Y obtuvieron un fallo para censurar totalmente el software reproductor de DVD. Prohibieron hasta linkar a una pagina web fuera de los EEUU en donde esta información estuviera legalmente disponible. Se ha hecho una apelación a este fallo. Me enorgullece decir que yo firmé un breve alegato en aquella apelación, aunque represento un papel bastante pequeño en esa batalla en particular.

El gobierno de los EEUU intervino directamente en favor del bando contrario. Esto no es sorprendente cuando consideras porqué la Digital Millennium Copyright Act fue aprobada en la primera votación. La razón es el sistema de financiación de campañas políticas que tenemos en EEUU, que es esencialmente un soborno legalizado, donde los candidatos son comprados por las empresas antes de ser siquiera elegidos. Y, por supuesto, ellos saben quién es su amo —saben para quién trabajan— y aprueban las leyes que les dan más poder a las compañías.

Qué ocurrirá con aquella batalla en particular, no lo sabemos. Pero mientras tanto Australia ha aprobado una ley similar y Europa está terminando de adoptar una: así que el plan es no dejar lugar en la Tierra donde esta información esté disponible para el público. Sin embargo, los EEUU siguen siendo el líder mundial en tratar de impedir que el público distribuya información que ha sido publicada.

[4] Ahora hay muchos paquetes de ese estilo. El primero se llamaba «DeCSS».

Los EEUU, sin embargo, no son el primer país en hacer de esto una prioridad. La Unión Soviética trató este tema como algo muy importante. Allí, la copia y redistribución no autorizadas eran conocidas como *samizdat* y para erradicarlas desarrollaron una serie de métodos: primero, guardias vigilando cada equipo de copia para verificar qué es lo que copiaba la gente e impedir hacer copias prohibidas. Segundo, duros castigos para cualquiera que pescaran haciendo copias prohibidas. Te podían mandar a Siberia. Tercero, buscar informantes, pidiéndole a todo el mundo que delate a sus vecinos y compañeros a la policía de la información. Cuarto, responsabilidad colectiva: «¡Tú! ¡Tú vas a vigilar a ese grupo! Si pesco a cualquiera de ellos haciendo copias prohibidas, irás a prisión. Así que vigílalos bien». Y quinto, propaganda, empezando en la niñez para convencer a todos de que sólo un horrible enemigo del pueblo podría perpetrar este copia prohibido.

Los EEUU están usando todos estos métodos ahora. Primero, guardias vigilando los equipamientos de copia. En tiendas de copiado, tienen guardias que verifican lo que copias. Pero emplear guardias humanos para vigilar qué copias en tu computadora sería demasiado caro; el trabajo humano es demasiado caro. Entonces tienen guardias robot. Ese es el propósito de la Digital Millennium Copyright Act. Este software va en tu ordenador; es la única manera en que puedes acceder a cierta información y te impide copiarla.

Ahora existe un plan para introducir este software en cada disco duro, de modo que habría archivos en tu disco a los que ni siquiera podrías acceder, excepto obteniendo permiso de acceso de algún servidor de red. Esquivar este software o aun decirle a otra gente cómo esquivarlo constituye un delito.

Segundo, duros castigos. Hace unos pocos años, si hacías copias de algo y se las entregabas a tus amigos, sólo para ayudarlos, esto no era un delito; nunca había sido un delito en los EEUU. Entonces lo convirtieron en una fechoría, de modo que te pueden poner en prisión durante años por compartir con tu vecino.

Tercero, informantes. Bueno, habréis visto los anuncios en la televisión, los anuncios en el metro de Boston pidiéndole a la gente que delate a sus compañeros de trabajo a la policía de la información, que oficialmente se llama Asociación de Editores de Software.

Y cuarto, responsabilidad colectiva. En los EEUU, se ha hecho mediante el alistamiento de los proveedores de Internet, haciéndolos legalmente responsables de todo lo que sus clientes publiquen. El único modo en que pueden evitar ser considerados responsables es si siguen invariablemente el procedimiento de desconectar o retirar la información en menos de dos semanas después de recibir una queja. Hace unos pocos días, oí que un sitio que contenía una protesta inteligente criticando al City Bank por algunas de sus malvadas políticas fue desconectado de esta manera. Hoy en día, ni siquiera te juzgan; tu sitio sencillamente es ser desconectado.

Y, finalmente, propaganda, comenzando desde la infancia. Para eso se usa la palabra «pirata». Si hacéis memoria, hace apenas unos pocos años el término «pirata» se aplicaba a los editores que no pagaban al autor. Pero ahora se le ha dado la vuelta completamente. Ahora se aplica a los miembros del público que escapan al control del editor. Está siendo usado para convencer a la gente de que sólo un malvado enemigo

del pueblo podría practicar la copia prohibida. Dice que «compartir con el prójimo es el equivalente moral de atacar un barco». Espero que no estéis de acuerdo, y si no lo estáis, espero que rehuséis a usar la palabra de tal manera.

Así que los editores están comprando leyes para darse más poder a sí mismos. Además, están extendiendo los plazos de duración del copyright. La Constitución de los EEUU dice que el copyright debe durar durante un tiempo limitado, pero los editores quieren que el copyright dure para siempre. Sin embargo, obtener una enmienda constitucional sería bastante difícil, así que encontraron una manera más fácil de lograr el mismo resultado. Cada veinte años extienden retroactivamente el copyright por veinte años mas. Así, el resultado es que, en un determinado momento, el copyright dura nominalmente por un cierto período y cualquier copyright dado va a expirar nominalmente algún día. Pero esa expiración nunca se alcanzará porque cada copyright se extenderá por veinte años, cada veinte años; entonces, ningún trabajo entrará en el dominio público otra vez. Este ha sido llamado el «plan del copyright perpetuo a plazos».

La ley que en 1998 extendió el copyright por 20 años se conoce como Ley de Mickey Mouse de Extensión del Copyright[5] porque uno de los principales auspiciantes de esta ley fue Disney. Disney se dio cuenta de que el copyright sobre Mickey Mouse iba a expirar, y ellos no quieren que eso ocurra nunca, pues hacen un montón de dinero con ese copyright.

Globalización

Ahora bien, el título original de esta charla era supuestamente «Copyright y Globalización». Si observáis la globalización, veréis que está compuesta de un conjunto de políticas que se hacen en nombre de la eficiencia económica, los llamados tratados de libre comercio, los cuales realmente están diseñados para darle a las compañías poder sobre las leyes y las directrices políticas. No son realmente tratados sobre libre comercio. Tienen que ver con transferencias de poder: retirar el poder de decidir leyes a los ciudadanos de cualquier país que pudieran acaso tener en cuenta sus propios intereses y dar ese poder a las compañías que no tendrán en cuenta los intereses de esos ciudadanos.

Desde su punto de vista la democracia es el problema y estos tratados están diseñados para terminar con el problema. Por ejemplo, el NAFTA [Zona de Libre Comercio de Norte América] de hecho contiene disposiciones, creo, que permiten a las compañías demandar a otro gobierno para así librarse de una ley que ellas piensen que dificulta sus ganancias en otro país. De este modo, las compañías extranjeras tienen más poder que los ciudadanos del país.

Existen pretensiones de extender esto más allá del NAFTA. Por ejemplo, este es uno de los objetivos de la así llamada Área de Libre Comercio de las Américas, extender este principio a todos los países de Sudamérica y el Caribe, y el acuerdo multilateral sobre la inversión iba a intentar diseminarlo por todo el mundo.

[5] La denominación oficial es «Ley de Sonny Bono para la extensión de la duración del copyright».

Una cosa que hemos visto en la década de 1990 es que estos tratados empiezan a imponer la legislación de copyright en todo el mundo, de maneras más poderosas y restrictivas. Estos tratados no son tratados de libre comercio. Son de hecho tratados de comercio controlado por empresas, usados para darle control a las corporaciones sobre el comercio mundial, para eliminar el libre comercio.

Cuando los EEUU eran un país en desarrollo en el siglo XIX, no reconocían los copyrights extranjeros. Ésta era una decisión tomada cuidadosamente y era una decisión inteligente. Se entendía que, para los EEUU, reconocer copyrights extranjeros sería desventajoso, que el dinero sería absorbido desde fuera y no haría mucho bien.

La misma lógica sería aplicable hoy día a los países en desarrollo, pero los EEUU tienen suficiente poder para obligarlos a ir en contra de sus intereses. De hecho, es un error hablar de los intereses de los países en este contexto. De hecho, estoy seguro de que la mayoría de ustedes han oído la falacia de intentar juzgar el interés público mediante la suma de la riqueza de todos. Si los trabajadores norteamericanos perdieran mil millones de dólares y Bill Gates ganase dos mil millones, los norteamericanos ¿estarían en general mejor? ¿Sería bueno para EEUU? Si miramos sólo el total parece que es bueno. Sin embargo, en realidad este ejemplo muestra que sumar el total es una manera incorrecta de juzgar, pues Bill Gates no necesita realmente otros dos mil millones pero la pérdida de mil millones por parte de otra gente que no tiene tanto puede, para empezar, ser dolorosa. Bien, en una discusión acerca de cualquiera de estos tratados de comercio, cuando oyes a la gente hablar de los intereses de este o de aquel país, lo que están haciendo con cada país es sumar los ingresos del total de población. Se suma el dinero de la gente rica y de la gente pobre. Así que aplicar esa misma falacia es en realidad una excusa para hacerte ignorar el efecto de la distribución de la riqueza en el país y en qué medida va a aumentar esa disparidad, como ha ocurrido en los EEUU.

Por lo tanto, no son realmente los intereses de los EEUU lo que se está defendiendo al imponer la legislación de copyright alrededor del mundo. Son los intereses de ciertos propietarios de empresas, muchos de los cuales están en los EEUU y algunos están en otros países. En ningún caso se defiende el interés público.

Repensar el copyright

Pero ¿qué cosa tendría sentido hacer? Si creemos en el objetivo del copyright declarado, por ejemplo, en la Constitución de los EEUU, que es el objetivo de promover el progreso, ¿qué *normas* sería inteligente usar en la era de las redes informáticas? Claramente, en vez de incrementar los poderes del copyright, tenemos que disminuirlos tanto como para darle al público cierto espacio de libertad donde pueda hacer uso de los beneficios de la tecnología digital, hacer uso de sus redes informáticas. Pero ¿hasta dónde podemos llegar con eso? Es una pregunta interesante porque no creo que debamos abolir totalmente el copyright. La idea de intercambiar algunas libertades a cambio de más progreso todavía podría ser ventajosa a cierto nivel, aun cuando el copyright tradicional restringe demasiada libertad. Pero para pensar acerca de esto de forma inteligente, lo primero que debemos reconocer es que no hay razón para hacerlo de modo

totalmente uniforme. No hay razón para insistir en conceder el mismo trato en todo tipo de trabajos.

De hecho, no es ese el caso actualmente porque hay un montón de excepciones para la música. La música es tratada de formas muy diferentes bajo la legislación de copyright. Pero la insistencia arbitraria en la uniformidad es usada astutamente por los editores. Eligen algún caso especial peculiar y argumentan que, en ese caso especial, sería ventajoso tener ciertas prerrogativas de copyright. Y luego dicen que en aras de la uniformidad, tiene que haber estas prerrogativas para todo. Entonces, por supuesto, eligen el caso especial en donde puedan hacer la argumentación más fuerte, aun cuando sea un caso especial poco frecuente y no muy importante después de todo.

Pero quizás deberíamos tener esas prerrogativas para ese caso especial en concreto. No tenemos que pagar el mismo precio para todo lo que compramos. Mil dólares por un coche nuevo puede ser un muy buen trato. Cien dólares por una botella de leche es un trato horrible. No pagarías el precio especial por cualquier cosa que compres en otros ámbitos de la vida. ¿Por qué hacerlo aquí?

Así que necesitamos observar las diferentes clases de obras, quisiera proponer una manera de hacerlo.

La primera clase son las obras funcionales —es decir, las que se usan para hacer un trabajo.

Esto incluye recetas, programas informáticos, manuales y libros de texto, obras de consulta como diccionarios y enciclopedias. Para todas estas obras funcionales creo que los problemas son básicamente los mismos que para el software y se pueden aplicar las mismas conclusiones. La gente debería tener la libertad aún de publicar una versión modificada porque es muy útil modificar trabajos funcionales. Las necesidades de la gente no son las mismas para todos. Si yo escribiera tal obra para hacer el trabajo que pienso que es necesario, tu idea sobre el trabajo que es necesario puede ser algo diferente. Entonces querrás modificar esta obra para que haga aquello que es bueno para ti. En ese punto, puede haber otra gente que tenga las mismas necesidades que tú y tu versión modificada puede ser buena para ellos. Todos los que saben cocinar saben esto y lo han sabido durante cientos de años. Es normal hacer copias de recetas y dárselas a otra gente y también es normal cambiar una receta. Si cambias la receta y cocinas para tus amigos y a ellos les gusta lo que están comiendo, podrán decirte «¿me puedes pasar la receta?». Entonces a lo mejor les apuntas tu versión y les das una copia. Esto es exactamente lo mismo que, mucho después, nosotros hemos empezado a hacer en la comunidad del software libre. Este es un tipo de obra.

El segundo tipo son las obras cuyo propósito es decir lo que cierta gente piensa. Su propósito es la opinión de cierta gente. Esto incluye, por ejemplo, memorias, artículos de opinión, publicaciones científicas, ofertas de compra y venta, catálogos de artículos para vender. La idea de estos trabajos es decirte qué es lo que alguien piensa, o qué vio, o qué cree. Modificarlos sería tergiversar a los autores; así que modificar estos trabajos no es una actividad socialmente útil. De este modo, la copia textual es lo único que realmente necesita la gente que le esté permitido hacer.

La siguiente pregunta es: ¿debería la gente tener derecho a hacer copias textuales con fines comerciales? ¿O es suficiente con las no comerciales? Veréis, son dos actividades diferentes que podemos distinguir, así que podemos tener en cuenta las preguntas por separado: el derecho a hacer copias textuales no comerciales y el derecho a hacer copias textuales comerciales. Bien, podría ser una buena política de compromiso tener un copyright que proteja la copia textual comercial pero permitir a todos el derecho a la copia textual no comercial. De esta manera, el copyright sobre la copia textual comercial, así como sobre todas las versiones modificadas —sólo el autor podría aprobar una versión modificada— seguiría proporcionando el mismo flujo de ganancias que provee ahora para costear la escritura de estos trabajos, en cualquier grado que sea.

Permitir la copia textual no comercial significa que el copyright ya no tendrá que entrometerse en el hogar de cada uno. Se convierte de nuevo en una regulación industrial, fácil de hacer cumplir e inofensiva. Ya no requerirá castigos draconianos ni informantes en pos de su cumplimiento. De este modo, obtendremos la mayor parte del actual beneficio —y evitaremos la mayor parte del horror del actual sistema.

La tercera categoría son las obras estéticas o de entretenimiento, donde lo más importante es la sensación de apreciar la obra. Para estas obras, la cuestión de la modificación es muy complicada porque, por un lado, está la idea de que estos trabajos reflejan la visión de un artista y cambiarlos es distorsionar esa visión. Por otro lado, tenemos el hecho de que se da un proceso popular, donde una sucesión de personas modificando una obra puede, a veces, producir un resultado que es extremadamente rico. Aún cuando existan artistas produciendo obras, tomar prestado de obras anteriores es a menudo muy útil. Algunas de las piezas teatrales de Shakespeare usaron historias tomadas de otras. Si las leyes de copyright de hoy en día hubieran tenido efecto entonces, esas obras teatrales hubieran sido ilegales. Así que es una cuestión difícil la de qué deberíamos hacer acerca de publicar versiones modificadas de una obra estética o artística, y podríamos tener que buscar más subdivisiones de la categoría para resolver este problema. Por ejemplo, puede ser que el entorno de los juegos de ordenador deba ser tratado de una manera; quizás todo el mundo debería ser libre de publicar versiones modificadas. Pero quizás una novela debería ser tratada de manera diferente; quizás, para ello, la publicación comercial requiera un acuerdo con el autor original.

Ahora bien, si la publicación comercial de estos trabajos estéticos se protegiera con el copyright, eso ocasionaría que buena parte del flujo de ganancias, que existe hoy en día, se dedicara a apoyar a los autores y músicos, y esto en el limitado grado en que el actual sistema los apoya, ya que lo hace muy mal. De este modo, este sería un compromiso razonable, justamente como en el caso de las obras que representan el punto de vista de una determinada persona.

Si miramos hacia adelante, al tiempo en el que la era de las redes de ordenadores haya comenzado plenamente, una vez que hayamos superado esta etapa de transición, podemos imaginar otra manera por la que los autores consigan dinero por su trabajo. Imaginemos que tenemos un sistema de dinero digital que te permite obtener dinero por tu trabajo. Imaginemos que tenemos un sistema de dinero digital que te permite enviar dinero a alguien a través de Internet. Esto puede hacerse de varias maneras; usando

cifrado, por ejemplo. E imaginemos que la copia textual de estos trabajos estéticos está permitida. Sin embargo están escritos de tal manera que cuando estás escuchando, o leyendo, o mirando uno de ellos, aparece una caja, a un lado en tu pantalla, que dice «haga click aquí para enviarle un dólar al autor», o al músico, o lo que sea. Y simplemente permanece ahí. No se interpone en tu camino. Está al lado. No interfiere contigo, pero está ahí, recordándote que es algo bueno apoyar a los escritores y a los músicos.

Así que si te gusta el trabajo que estás leyendo o escuchando, eventualmente dirás: «¿Por qué no he de darle a esta gente un dólar? Es sólo un dólar. ¿Qué es eso? Ni siquiera lo notaré». Y las personas empezarán a enviar un dólar. Lo bueno de esto es que hace de la copia el aliado de los autores y de los músicos. Cuando alguien le envía por correo electrónico a un amigo una copia, ese amigo podría enviar un dólar también. Si realmente te gusta, podrías enviar un dólar más de una vez y ese dólar es más de lo que obtienen hoy si compras un libro o compras un CD, ya que obtienen una minúscula fracción de la venta. Los mismos editores que están exigiendo pleno poder sobre el público en nombre de los autores y músicos, están aprovechándose todo el tiempo de esos mismos autores y esos mismos músicos.

Os recomiendo leer el artículo de Courtney Love en la revista *Salon*, un artículo sobre los piratas que planean usar el trabajo de los músicos sin pagarles. Estos piratas son las compañías discográficas que les pagan un promedio del 4 % de las ventas. Por supuesto, los músicos de mayor éxito reciben una porción mayor. Obtienen más del 4 % de sus grandes ventas, lo que significa que la gran mayoría de los músicos que tienen un contrato discográfico obtienen menos del 4 % de sus pequeñas ventas.

Éste es el modo en que funciona: la compañía discográfica gasta dinero en publicidad y considera este gasto como un adelanto a los músicos, aunque los músicos nunca lleguen a verlo. De este modo, nominalmente, cuando compras un CD, cierta fracción de ese dinero va a los músicos, pero realmente no es así. En realidad, está destinado a pagar los gastos publicitarios y solamente si los músicos son de gran éxito podrán ver algo de ese dinero.

Los músicos, por supuesto, firman sus contratos discográficos porque tienen la esperanza de ser uno de esos pocos que se hacen ricos. Así que, esencialmente, es una lotería que se ofrece a los músicos para tentarlos. Aun cuando sean buenos músicos, pueden no ser buenos para razonar de forma lógica y cuidadosa y de esta forma poder ver esta trampa. De este modo, firman y probablemente todo lo que obtienen es publicidad. Bueno, ¿por qué no les damos publicidad de una manera diferente? No a través de un sistema basado en la restricción del público, un sistema de los complejos industriales que nos entristece con una música malísima que es fácil de vender. En cambio, ¿por qué no hacer del impulso natural del oyente a compartir la música que le gusta el aliado de los músicos? Si tenemos esta caja que aparece en el reproductor como un modo de enviar un dólar a los músicos, las redes informáticas podrían ser el mecanismo para dar a los músicos esta publicidad, la misma publicidad que es todo lo que ahora obtienen de los contratos discográficos.

Debemos reconocer que el sistema de copyright existente hace un pésimo trabajo de apoyo a los músicos. Tan malo como el que hace el comercio mundial al intentar elevar

el nivel de vida en Filipinas y en China. Esas zonas industriales donde todo el mundo trabaja en *sweatshops* y todos los productos se hacen *sweatshops.* La globalización es una manera muy ineficiente de elevar el nivel de vida de los pueblos de ultramar. Pongamos por caso, a un norteamericano se le paga veinte dólares la hora para hacer algo y le das ese trabajo a un mexicano a quien se le paga quizás seis dólares por día. Lo que ocurre aquí es que tomas una gran cantidad de dinero de un trabajador norteamericano, le das una fracción minúscula, un pequeño porcentaje, a un trabajador mexicano, y el resto se lo devuelves a la compañía. De modo que si tu meta es elevar el nivel de vida de los trabajadores mexicanos, esta es una pésima manera de hacerlo.

Es interesante ver cómo el mismo fenómeno se da en la industria del copyright, la misma idea general. En nombre de estos trabajadores, quienes ciertamente merecen algo, proponen medidas que les dan una diminuta porción y en realidad aumentan el poder de las compañías para controlar nuestras vidas.

Si estás tratando de reemplazar un sistema muy bueno, tienes que hacer un esfuerzo muy grande para encontrar una alternativa mejor. Si sabes que el actual sistema es deplorable, no es tan difícil encontrar una alternativa mejor; el patrón de comparación es hoy muy bajo. Debemos recordarlo siempre esto cuando consideramos cuestiones relativas a la política del copyright.

Creo que dije la mayor parte de lo que quiero decir. Quisiera mencionar que mañana es el *phone-in sick day*[6] en Canadá. Mañana se da inicio a una cumbre para terminar de negociar el Área de Libre Comercio de las Américas, con el fin de extender el poder de las corporaciones a un mayor número de países; se está planeando una gran protesta en Quebec. Hemos visto métodos extremos para aplastar esta protesta. Se está impidiendo a muchos estadounidenses la entrada a Canadá a través de la frontera que, se supone, debería permitirles entrar en cualquier momento. Bajo las excusas más endebles han construido un muro alrededor del centro de Quebec para usarlo como fortaleza a fin mantener a los manifestantes fuera. Hemos visto gran cantidad de trucos sucios usados contra la manifestación pública en contra de estos tratados. De este modo, cualquier brizna de democracia que nos quede después de que se le haya retirado a nuestros gobernantes democráticamente electos el poder de gobernar y después de que se le haya dado a las compañías y a los organismos internacionales no electos, lo que sea que quede después de eso, puede que no sobreviva a la supresión de la protesta pública contra esa tendencia.

He dedicado diecisiete años de mi vida a trabajar en el software libre y en asuntos relacionados con él. No lo he hecho porque piense que es la cuestión política más importante del mundo. Lo hice porque era el área en donde vi que tendría que usar mejor mis destrezas para hacer el mayor bien. Pero lo que ha ocurrido es que las cuestiones políticas en general han evolucionado y la cuestión política más importante del mundo, hoy, es resistir la tendencia a dar poder a las compañías en detrimento del público y de los gobiernos. Veo el software libre y los asuntos relacionados a la información

[6]Evento del movimiento global que podría traducirse como «día que toca ponerse enfermo», en alusión a la excusa usada para no tener que ir al trabajo y poder asistir en su lugar a la manifestación. El evento al que se refiere Stallman tuvo lugar el 20 de abril de 2001. [*N. del E.*]

como parte de esa cuestión de primer orden. Así que me he encontrado indirectamente trabajando en este problema. Espero contribuir en algo a ese esfuerzo.

Turno de preguntas

THORNBURN. En un momento vamos a pasar a escuchar las preguntas y comentarios del público. Pero antes permítanme ofrecerles una breve y somera intervención. Me parece que la guía práctica más fuerte e importante que Stallman nos ofrece tiene dos elementos clave. Uno es el reconocimiento de que las viejas suposiciones sobre el copyright, los viejos usos del copyright, son inapropiados; están siendo desafiados o socavados por el advenimiento del ordenador y de las redes informáticas. Eso puede ser obvio, pero es esencial.

El segundo es el reconocimiento de que la era digital nos obliga a reconsiderar cómo distinguimos y sopesamos las formas del trabajo intelectual y creativo. Stallman, indudablemente, está en lo cierto al afirmar que ciertos tipos de iniciativas intelectuales justifican más protección mediante el copyright que otras. Tratar de identificar sistemáticamente estos diferentes tipos o niveles de protección por medio del copyright me parece una valiosa manera de ocuparse de los problemas relativos al trabajo intelectual representados por el advenimiento del ordenador.

Pero creo que estoy detectando otro tema subyacente en lo que Stallman ha estado diciendo y que no está en realidad directamente relacionado con los ordenadores, sino más ampliamente con cuestiones de autoridad democrática y con el poder creciente que los gobiernos y las empresas ejercen sobre nuestras vidas. Este lado populista y anticorporativo del discurso de Stallman es enriquecedor pero también reduccionista, potencialmente simplista. Y es también quizás demasiado idealista. Por ejemplo, cómo podría un novelista o un poeta o un autor de canciones o un músico o el autor de un libro de texto académico, sobrevivir en este mundo feliz en que la gente es alentada pero no obligada a pagar a los autores. En otras palabras, me parece que la brecha entre la práctica existente y las posibilidades visionarias sobre las que especula Stallman, es todavía inmensamente ancha.

Entonces voy a concluir preguntándole a Stallman si quisiera desarrollar un poco ciertos aspectos de su charla y, específicamente, si es que tiene más ideas sobre la manera en que aquellos que llamaremos «creadores tradicionales» pueden ser protegidos bajo su sistema de copyright.

STALLMAN. Primero de todo, tengo que señalar que no debemos usar el término «protección» para describir lo que hace el copyright. El copyright restringe a la gente. El término «protección» es un término propagandístico que usan las empresas propietarias de copyright. El término «protección» significa impedir que algo sea, de alguna manera, destruido. Bien, yo no creo que una canción se destruya porque se escuchen más copias de ella. Tampoco creo que una novela se destruya si más gente está leyendo copias de ella. Así que no usaré esa palabra. Pienso que conduce a la gente a identificarse con el bando equivocado.

También, es una muy mala idea pensar acerca de la «propiedad intelectual» por dos razones: primero, prejuzga la pregunta más fundamental en este ámbito, que es: ¿cómo deberían ser tratadas estas cosas, deberían tratarse como un tipo de propiedad? Usar el término «propiedad intelectual» para describir la cuestión es presuponer que la respuesta es «sí», que ésa es la manera de tratar las cosas y no de otra manera.

Segundo, promueve la sobregeneralización. La propiedad intelectual es un término genérico utilizado para varios sistemas legales con orígenes independientes como el copyright, las patentes, las marcas registradas, los secretos comerciales y también algunas otras cosas más. Se trata de cosas casi completamente diferentes; no tienen nada en común. Pero a la gente que oye el término «propiedad intelectual» se le lleva hacia una falsa imagen, creen que hay un principio general de propiedad intelectual que es aplicado en distintas áreas específicas. De este modo, asumen que esas diferentes áreas de la ley son similares. Esto lleva no sólo a un pensamiento confuso acerca de qué es correcto hacer; lleva a la gente a no poder entender qué es lo que de hecho dice la ley, ya que suponen que la ley de copyright, la ley de patentes y la ley de marcas registradas son similares, cuando, de hecho, son totalmente diferentes.

Así que si queréis promover el pensamiento cuidadoso y el entendimiento claro de qué es lo que la ley dice, evitad el uso del término «propiedad intelectual». Hablad de copyright. O hablad de patentes. O hablad de marcas registradas o cualquiera que sea el asunto del que queráis hablar. Pero no habléis de propiedad intelectual. Una opinión sobre propiedad intelectual casi tiene que ser necesariamente absurda. Yo no tengo una opinión acerca de la propiedad intelectual. Tengo opiniones acerca del copyright, las patentes y las marcas registradas, y son diferentes. Llegué a ellas a través de procesos de pensamiento diferentes porque esos sistemas legales son totalmente diferentes.

De todos modos, he hecho una digresión, pero es terriblemente importante.

Ahora permitidme ir al grano. Por supuesto, no podemos ver ahora cómo podría funcionar correctamente, o si es que podría funcionar, pedirle a la gente que pague dinero de forma voluntaria a los autores y músicos que les gustan. Una cosa obvia es que todo lo bien que puede llegar a funcionar un sistema así es proporcional al número de personas que participan de la red, y ese número, lo sabemos, se incrementará exponencialmente dentro de unos años. Si lo intentásemos hoy, podría fallar y ello no probaría nada porque con diez veces más gente participando podría funcionar.

La otra cosa es que aun no tenemos este sistema de desembolso de dinero digital. Así que en realidad no podemos intentarlo hoy. Podría intentarse algo un poco parecido. Puedes contratar servicios en los que puedes pagarle dinero a alguien —cosas como Pay Pal. Pero, antes de poder pagarle a nadie mediante Pay Pal, tendrás que soportar un complejo papeleo y darles información personal sobre ti. Y ellos hacen registros de a quiénes les pagas. ¿Puedes confiar en que no harán un mal uso de esta información?

El dólar puede no desalentarte, pero las dificultades que acarrea el sistema de pago sí que pueden. La idea es que debería ser tan fácil como quitarle un caramelo a un niño pagar cuando urja la necesidad, de modo que no haya nada que te desaliente excepto la cantidad de dinero. Y si es lo bastante pequeño, no tiene porque desalentarte. Sabemos, en cambio, que los *fans* pueden realmente amar a los músicos y sabemos que alentar

a los *fans* a copiar y redistribuir música ha sido hecho por algunas bandas que fueron, y son, bastante exitosas, como los Grateful Dead. Ellos no tuvieron problemas para ganarse la vida con su música por haber alentado a sus *fans* a grabar y copiar cintas. Ni siquiera bajaron las ventas de sus discos.

Nos estamos desplazando gradualmente de la era de la imprenta a la era de las redes informáticas, pero esto no ocurre de un día para otro. La gente todavía compra montones de discos y eso continuará pasando probablemente durante muchos años — quizás para siempre. En tanto eso continúe, simplemente con tener un copyright que se aplique a las ventas comerciales de discos se podría hacer un trabajo tan bueno de apoyo a los músicos como el que se hace hoy en día. Por supuesto, no es muy bueno, pero, al menos, la cosa no irá a peor.

PREGUNTA. [Un comentario y una pregunta acerca de la libre descarga y acerca del intento de Stephen King[7] de comercializar una de sus novelas a través de la web.]

STALLMAN. Sí, es interesante saber qué hizo y qué ocurrió. Cuando al principio oí hablar sobre aquello, estaba encantado. Pensé: tal vez está dando un paso hacia un mundo que no esté basado en tratar de mantener al público apresado por una cadena de hierro. Entonces vi que de hecho había escrito para pedirle a la gente que pague. Explicando lo que hizo, estaba publicando una novela como una serie, por entregas, y dijo: «Si obtengo suficiente dinero, entregaré más». Pero la petición que escribió era a duras penas una petición. Era una afrenta al lector. Decía: «Si no pagáis, es que sois malvados. Y si hay demasiados de vosotros que sois malvados, entonces yo simplemente dejaré de escribir esto».

Bien, claramente, ésa no es la manera de hacer que el público sienta ganas de enviarte dinero. Tienes que hacer que te amen, no que te teman.

PREGUNTA. [El mismo miembro del público] Los detalles fueron que él pidió que cierto porcentaje —no sé el porcentaje exacto, alrededor de 90 % me parece— de gente le enviara cierta cantidad de dinero, la cual era, creo, uno o dos dólares, o algo de esa magnitud. Tenías que escribir tu nombre y tu dirección de correo electrónico y alguna otra información para descargar la novela y si después del primer capítulo ese porcentaje no se alcanzaba, dijo que no publicaría otro. Fue muy hostil con el público que la descargaba.

PREGUNTA. ¿No está abierto al abuso de la gente, mediante el plagio, este sistema en donde no hay copyright pero en el que se le pide a la gente que haga donaciones voluntarias?

STALLMAN. No. Eso no es lo que he propuesto. Recuerda, estoy proponiendo que debería haber copyright cubriendo la distribución comercial y permitiendo sólo una redistribución de forma no comercial. Así que cualquiera que modifique la obra agregándole

[7] El escritor de novelas de terror Stephen King intentó vender uno de sus libros a través de la red en una serie de descargas (podías comprar un capítulo cada vez), pero cerró el sistema antes de acabar de escribir el libro.

un contador a su sitio web, en lugar del contador del sitio web del verdadero autor, estaría infringiendo el copyright y podría ser demandado exactamente como podría ser demandado hoy.

PREGUNTA. Ya veo. ¿Entonces todavía imaginas un mundo en el que hay copyright?

STALLMAN. Sí. Como he dicho, para esa clase de obras. No estoy diciendo que todo debería estar permitido. Estoy proponiendo reducir los poderes del copyright, no abolirlos.

THORNBURN. Una pregunta que se me ocurrió mientras hablabas, Richard, y otra vez mientras respondías a esta pregunta: ¿por qué no consideras las maneras en que el ordenador, por sí mismo, elimine completamente a los intermediarios —del modo en que Stephen King se negó a hacer— y pueda establecer una relación personal?

STALLMAN. Bien, pueden y, de hecho, esta donación voluntaria es una manera...

THORNBURN. ¿Piensas que ello no involucrará al editor en ningún caso?

STALLMAN. En absoluto. Espero que no lo haga, verás, porque los editores explotan a los autores terriblemente. Cuando les preguntas a los representantes de los editores acerca de esto, dicen: «Bien, sí, si un autor o una banda no desea pasar por nosotros, no debería estar legalmente obligado a pasar por nosotros». Pero, de hecho, ellos hacen todo lo que pueden para impedir que eso resulte factible. Por ejemplo, están proponiendo formatos de copia restringida, de modo que para publicar en esos formatos, tendrías que pasar por los grandes editores, ya que no les dirán a nadie más cómo hacerlo. De este modo, su esperanza es un mundo en donde los reproductores reproduzcan esos formatos, y para obtener cualquier cosa que puedas reproducir en esos reproductores habrá que pasar por los editores. Así que, de hecho, aunque no haya una ley que prohíba al autor o al músico publicar de forma directa, esto no será factible. Está también el señuelo de que tal vez te hagas rico. Dicen: «Te publicitaremos y quizás te vuelvas tan rico como los Beatles» —haceros a la idea de cualquier grupo de éxito— y, por supuesto, sólo una minúscula fracción de los músicos tendrá esa suerte. Pero pueden ser llevados así a firmar contratos que los encerrarán para siempre.

Los editores tienden a ser malévolos a la hora de respetar sus contratos con los autores. Por ejemplo, los contratos de libro habitualmente dicen que si un libro se agota, los derechos vuelven al autor, y los editores generalmente no han sido muy buenos ejemplificando esa cláusula. A menudo se les ha tenido que obligar a respetarla. Bien, lo que están empezando a hacer ahora es usar la publicación electrónica como una excusa para decir que nunca se agotará; así que nunca tendrán que devolver los derechos. Su idea es que cuando el autor está necesitado haz que firme, y, desde entonces, no tendrá poder; sólo el editor tiene el poder.

PREGUNTA. ¿Sería bueno tener licencias libres para distintos tipos de obras que protejan el derecho del usuario a copiar del modo en que sea apropiado para cada tipo de trabajo?

STALLMAN. Bien, hay gente trabajando en esto. Pero para trabajos no funcionales, una cosa no sustituye la otra. Observemos un tipo de trabajo funcional, digamos un procesador de texto. Bien, si alguien hace un procesador de texto libre, puedes usarlo; no necesitas los procesadores de texto no libres. Pero yo no diría que una canción libre puede sustituir a todas las canciones no libres o que una novela libre sustituya a todas las novelas no libres. Para esa clase de obras, es diferente. Entonces, lo que pienso que simplemente debemos hacer es reconocer que estas leyes no merecen ser respetadas. No es incorrecto compartir con tu vecino, y si alguien intenta decirte que no puedes compartir con tu vecino, no deberías escucharlo.

PREGUNTA. Con respecto a los trabajos funcionales, según tu manera de pensar ¿cómo se equilibra la necesidad de abolir el copyright con la necesidad de incentivos económicos para hacer que se desarrollen estos trabajos funcionales?

STALLMAN. Bien, lo que vemos es, antes que nada, que este incentivo económico es mucho menos necesario que lo que la gente ha estado suponiendo. Fijaros en el movimiento de software libre, donde tenemos más de cien mil voluntarios a tiempo parcial desarrollando software libre. También vemos que hay otras maneras de obtener dinero que no están basadas en impedir que el público copie y modifique estos trabajos. Ésa es la lección interesante del movimiento del software libre. Aparte del hecho de que te ofrece una manera en que puedes usar un ordenador y conservar tu libertad de compartir y cooperar con otra gente, también nos muestra que esta suposición negativa de que la gente nunca haría estas cosas a menos que se les den poderes especiales para forzar a la gente a pagarles, es sencillamente incorrecta. Mucha gente hará estas cosas. De este modo, si echas un vistazo a, digamos, la escritura de monografías que sirven como libros de texto en muchos campos de la ciencia, excepto los muy básicos, los autores no hacen dinero. Ahora tenemos un proyecto para hacer una enciclopedia libre, que en realidad es un proyecto comercial, y está progresando. Teníamos un proyecto para una enciclopedia GNU, pero lo fundimos con otro proyecto comercial cuando ellos adoptaron nuestra licencia. En enero, se pasaron a la licencia de documentación libre GNU para todos los artículos de su enciclopedia. Entonces dijimos: «Bien, unamos fuerzas con ellos y alentemos a la gente a contribuir». Se llama «Nupedia» y podéis encontrar un enlace a ella en http://www.gnu.org/encyclopedia. Así que hemos ampliado el desarrollo comunitario de una base libre de conocimientos útiles desde el software a una enciclopedia. Ahora estoy bastante seguro de que en todas estas áreas de obras funcionales, no necesitamos ese incentivo económico hasta el punto de que debamos arruinar el uso libre de esas obras.

THORNBURN. Bien, qué hay de las otras dos categorías —pensamientos personales y entretenimiento.

STALLMAN. Para los otros dos tipos de trabajo, no sé. No sé si la gente va a escribir algún día novelas sin preocuparse por ganar dinero con ello. En una sociedad post-escasez, pienso que sí. Puede que lo que necesitemos hacer, para alcanzar la sociedad post-escasez, es deshacernos del control empresarial sobre la economía y las leyes. Así que, en efecto, es como el problema del huevo y la gallina, como podéis ver. ¿Qué hace-

mos primero? ¿Cómo obtenemos un mundo en donde la gente no tenga que conseguir dinero desesperadamente, si no es eliminando el control empresarial? ¿Y cómo podemos eliminar el control empresarial a no ser que. . . ? De todos modos, no lo sé, pero por eso estoy tratando de proponer primero un sistema de copyright de compromiso, y, segundo, el pago voluntario apoyado por un sistema de copyright de compromiso como una manera de proveer un flujo de ganancias a la gente que escribe estos trabajos.

PREGUNTA. ¿Cómo esperas implementar este sistema de copyright de compromiso bajo la presión de los intereses corporativos sobre los políticos estadounidenses, dado su sistema de financiación de campañas?

STALLMAN. Me supera. Ojalá lo supiera. Es un problema terriblemente difícil. Si supiera cómo resolver ese problema, lo resolvería y nada en el mundo podría hacerme sentir más orgulloso.

PREGUNTA. ¿Cómo luchas contra el control empresarial? Porque cuando observas las sumas de dinero destinadas al lobby empresarial en los tribunales, es tremendo. Pienso que el caso DeCSS (Decryption of Contents Scrambling System), del que estás hablando, le está costando algo así como un millón y medio de dólares a la defensa. Dios sabe cuánto le está costando a la parte empresarial. ¿Tienes alguna idea de cómo hacerse cargo de estas enormes sumas de dinero?

STALLMAN. Tengo una sugerencia. Si aconsejáramos boicotear totalmente las películas, pienso que la gente ignoraría ese consejo. Podrían considerarlo demasiado radical. Así que quisiera hacer una sugerencia levemente diferente, que lleva a lo mismo, y es: no vayas al cine a menos que tengas una razón sustancial para creer que la película es buena. Ahora bien, esto conduciría en la práctica a casi el mismo resultado que el boicot total de las películas de Hollywood. En extensión es casi lo mismo, pero en intensidad es muy diferente. Hemos notado que mucha gente va al cine por razones que nada tienen que ver con lo buena, o no, que ellos piensen que es la película. Así que si cambias eso, si sólo vas a ver una película cuando tienes alguna razón sustancial para creer que es buena, les estarás quitando un montón de dinero.

THORNBURN. Una manera de entender todo este discurso hoy, pienso, es reconocer que siempre que las tecnologías radicales, potencialmente transformadoras, aparecen en la sociedad, hay una lucha sobre su control. Hoy estamos repitiendo lo que ocurrió en el pasado. De este modo, desde este ángulo, puede no haber una razón para la desesperanza, o incluso para el pesimismo, acerca de qué pueda ocurrir a largo plazo. Pero a corto plazo, las luchas por el control de textos e imágenes, y por el control de todo tipo de información, serán probablemente dolorosas y extensas. Por ejemplo, como profesor de comunicación, mi acceso a imágenes ha sido restringido, en años recientes, de un modo como nunca antes había ocurrido. Si escribo un ensayo en el que quiero usar imágenes fijas, incluso de películas, es mucho más difícil obtener permisos de uso, y los precios son mucho más elevados, aun cuando dé argumentos sobre la investigación

intelectual y la categoría legal de *fair use*[8]. De esta forma, pienso, en este momento de transformación, que las perspectivas a un plazo mayor pueden, de hecho, no ser tan perturbadoras como lo que está ocurriendo a corto plazo. Sin embargo, en cualquier caso, necesitamos comprender toda nuestra experiencia contemporánea como una versión renovada de la lucha por el control de los recursos tecnológicos, que es un principio recurrente de la sociedad occidental.

También es esencial entender que la historia de las viejas tecnologías es, en sí misma, una materia complicada. El impacto de la imprenta en España, por ejemplo, fue radicalmente diferente de su impacto en Inglaterra o en Francia.

PREGUNTA. Una de las cosas que me molesta cuando oigo discusiones sobre el copyright es que a menudo comienzan con: «Queremos un cambio de ciento ochenta grados. Queremos eliminar todo tipo de control». Me parece que parte de lo que subyace bajo las tres categorías que fueron sugeridas es un reconocimiento de que hay algún tipo de sabiduría en el copyright. Algunos de los críticos de la dirección que está tomando el copyright hoy, creen, de hecho, que debería ser respaldado y funcionar mucho más como las patentes y las marcas registradas en cuanto a su duración. Me pregunto si nuestro conferenciante querrá comentar esta estrategia.

STALLMAN. Estoy de acuerdo en que acortar el plazo de validez del copyright es una buena idea. No hay absolutamente ninguna necesidad, en términos de alentar la publicación, de que los copyrights puedan durar hasta ciento cincuenta años, lo que, en algunos casos, es posible bajo la presente ley. Ahora bien, las compañías estuvieron diciendo que un copyright de setenta y cinco años sobre un trabajo hecho por encargo no era lo bastante largo para hacer posible la producción de esos trabajos. Me gustaría desafiar a esas compañías a que presenten hojas de cálculo con balances proyectados para los próximos setenta y cinco años respaldando esa afirmación. Lo que ellos en realidad querían era, sencillamente, poder extender el copyright sobre los trabajos viejos, de modo que puedan seguir restringiendo su uso. Pero cómo se puede alentar una mayor producción en la década de 1920 extendiendo el copyright hoy, no lo sé, a menos que tengan una máquina del tiempo en algún lugar. Por supuesto, en una de sus películas, tenían una máquina del tiempo. Así que eso debe ser lo que afectó su forma de pensar.

PREGUNTA. ¿Has pensado en extender el concepto de uso razonable? Y, ¿hay alguna variación o distinción al respecto que quieras presentarnos?

STALLMAN. Bien, la idea de dar a todo el mundo permiso para hacer copias textuales no comerciales de dos tipos de obras, ciertamente puede ser pensada como una extensión de lo que es el uso razonable. Es mayor de lo que actualmente se reconoce como uso

[8]El *fair use*, que se suele traducir como «uso razonable» o «uso justo» sin que la traducción ayude a saber qué clase de uso es ese, es en realidad una figura jurídica del derecho anglosajón que equivale a aquellos usos de material sujetos a derechos de autor que no son controlados por la ley de copyright y que por tanto son excepciones a la misma. La Ley de Propiedad Intelectual española no contiene tal figura, y lo llama simplemente «excepciones», por ejemplo el préstamo en bibliotecas o la copia privada son ejemplos de *fair use* o «excepciones» tanto en EE.UU. como en España. [*N. del E.*]

razonable. Si tu idea es que el público cede ciertas libertades para obtener más progreso, entonces puedes trazar la línea en varios lugares diferentes. ¿Qué libertades cede el público y cuáles conserva?

PREGUNTA. Por seguir la conversación sólo un momento: en ciertos ámbitos del entretenimiento, existe el concepto de exhibición pública. Así, por ejemplo, el copyright no nos impide cantar villancicos en Navidad pero impide su ejecución pública. Y yo me pregunto si no sería útil, en lugar de expandir el uso razonable a la copia textual ilimitada, no comercial, pensar en algo menos que eso, pero más que el presente concepto de uso razonable.

STALLMAN. Yo pensaba que eso sería suficiente, y entonces Napster me convenció de lo contrario, porque Napster es utilizado por sus usuarios para redistribución textual, no comercial. El servidor Napster, en sí mismo, es una actividad comercial, pero la gente que de hecho está poniendo el material lo hace de manera no comercial, ellos podrían haberlo hecho en sus propios sitios web igual de fácilmente. La tremenda excitación y interés sobre el uso de Napster, muestra que eso es muy útil. Así que ahora estoy convencido de que la gente debería tener derecho a publicar copias textuales, no redistribuidas comercialmente, de cualquier cosa.

PREGUNTA. Una analogía que me fue sugerida recientemente para toda la cuestión de Napster es la analogía de la biblioteca pública. Supongo que algunos de ustedes, que han oído hablar sobre los argumentos a favor de Napster, han oído ya esta analogía. Me pregunto si podrías comentarla. Los defensores que dicen que Napster debería continuar y que no debería haber restricciones sobre él, a veces dicen algo como esto: «Cuando las personas van a la biblioteca pública y piden prestado un libro, no están pagando por él, y pueden pedirlo prestado docenas de veces, cientos de veces, sin cargo adicional. ¿Por qué Napster es diferente?».

STALLMAN. Bueno, no es exactamente lo mismo. Pero debería señalarse que los editores quieren transformar a las librerías públicas en tiendas en las que se paga por el uso. Así que están en contra de las librerías públicas.

PREGUNTA. ¿Pueden estas ideas sobre el copyright sugerir algunas ideas para ciertas cuestiones sobre la ley de patentes, tales como hacer medicamentos genéricos baratos para usar en África?

STALLMAN. No, no hay absolutamente ninguna similitud. Las cuestiones de patentes son totalmente distintas de las cuestiones relativas al copyright. La idea de que tienen algo que ver es una de las consecuencias desafortunadas de usar el término «propiedad intelectual» y alentar a la gente a asociar estas cuestiones, porque, como habéis oído, estuve hablando de cuestiones en las que el precio de una copia no es lo crucial. Pero, ¿cuál es el asunto central cuando se trata de medicamentos contra el SIDA para África? Es el precio, nada más que el precio.

Ahora bien, el tema del que estuve hablando surge porque la tecnología de información digital da a cada usuario la facultad de crear copias. Bien, no hay nada que nos dé la facultad de crear copias de medicamentos. No tengo la posibilidad de copiar un

medicamento que haya conseguido. De hecho, nadie puede; no es así como se hacen los medicamentos. Estos medicamentos sólo pueden hacerse en costosas fábricas y se hacen en costosas fábricas centralizadas, ya sean genéricos o importados de los EE.UU. De cualquier manera, se harán en un pequeño número de fábricas y las cuestiones son, simplemente, cuánto cuestan y si están disponibles a un precio que la gente en África pueda pagar.

Así que se trata de una cuestión tremendamente importante, pero es una cuestión totalmente diferente. Sólo existe un área en la que aparece una cuestión con las patentes que es de hecho similar a estas cuestiones de libertad de copia, y es en la agricultura. Dado que hay ciertas cosas patentadas que pueden ser copiadas, más o menos: las cosas vivientes. Se copian a ellas mismas al reproducirse. No es necesariamente una copia exacta; se remezclan los genes. Pero el hecho es que los granjeros durante milenios han estado haciendo uso de esta capacidad de las cosas vivientes de hacer copias de sí mismas. La agricultura es, básicamente, copiar las cosas que criaste y seguir copiándolas cada año. Cuando son patentadas variedades de plantas y animales, cuando los genes son patentados y usados en ellas, el resultado es que a los granjeros se les prohíbe hacer lo que venían haciendo.

Un granjero en Canadá tenía una variedad patentada creciendo en su campo y dijo: «Yo no lo hice deliberadamente. El polen voló, y esos genes se introdujeron entre mis plantas». Y se le dijo que eso no importaba; tuvo que destruirlas de todos modos. Éste es un ejemplo extremo de cómo puede el gobierno alinearse con un monopolista.

De este modo, creo que, siguiendo los mismos principios que aplico a la copia de cosas en un ordenador, los granjeros deberían tener un incuestionable derecho a guardar sus semillas y criar su ganado. Quizás puedas tener patentes cubriendo compañías vendedoras de semillas, pero no deberían cubrir a los granjeros.

PREGUNTA. Para tener un modelo exitoso hay que hacer mas cosas que tener sólo la licencia. ¿Puedes responder a eso?

STALLMAN. Totalmente. Bien, ya sabéis, no conozco las respuestas. Pero parte de lo que creo crucial para desarrollar información libre, funcional, es el idealismo. La gente tiene que reconocer que es importante para esta información ser libre, que cuando la información es libre, se puede hacer pleno uso de ella. Cuando está restringida, no puedes. Tienes que reconocer que la información no libre es un intento de dividir a la gente y mantenerla desamparada y con la cabeza agachada. En este punto, pueden llegar a la idea de: «Trabajemos juntos para producir la información que queremos usar, de modo que no esté bajo el control de alguna persona poderosa que pueda dictarnos qué es lo que podemos hacer».

Esto nos impulsa con fuerza. No sé cómo va a funcionar en la distintas áreas, pero pienso que en el ámbito de la educación, buscando libros de texto, creo ver una manera en que puede hacerse. Hay un montón de docentes en el mundo, docentes que no están en universidades prestigiosas —quizás están en la escuela secundaria, quizás en la preparatoria— donde no escriben ni publican gran cosa y no existe una tremenda demanda sobre ellos. Pero muchos de ellos son inteligentes. Muchos de ellos conocen sus

materias bien y podrían escribir libros de texto sobre montones de temas y compartirlos con la gente, recibir un enorme aprecio por la gente que aprende de ellos.

PREGUNTA. Eso es lo que decía. Pero lo curioso es que conozco la historia de la educación. Eso es lo que hago: proyectos educativos con medios electrónicos. No podría encontrar un ejemplo. ¿Conoces alguno?

STALLMAN. No, no conozco. Empecé proponiendo esta enciclopedia libre y base de aprendizaje hace un par de años y pensé que podría llevar probablemente una década lograr que las cosas comenzaran a rodar. Ahora ya tenemos una enciclopedia en marcha. Así que las cosas van más rápido de lo que esperaba. Pienso que lo que se necesita es que unas pocas personas empiecen a escribir libros de texto libres. Escribir uno sobre cualquiera que sea tu tema favorito o una fracción de uno. Escribe unos pocos capítulos de uno y desafía a otras personas a escribir el resto.

PREGUNTA. De hecho, lo que yo buscaba era algo todavía mejor que eso. Lo importante es que alguien cree una infraestructura en la que todos los demás puedan contribuir. No hay ninguna infraestructura pensada para primaria y secundaria, en ningún lugar, para poder contribuir con material.

Puedo obtener información de muchos lugares pero no está disponible bajo licencias libres, así que no puedo usarla para hacer un libro de texto libre.

STALLMAN. En realidad, el copyright no cubre los hechos. Sólo cubre el modo en que están escritos. Así que puedes aprender un montón de cualquier lugar y después escribir un libro de texto, y puedes hacer ese libro de texto libre, si quieres.

PREGUNTA. Pero yo no puedo escribir por mí mismo todos los libros de texto que un estudiante necesita para cursar en la escuela.

STALLMAN. Bien, es verdad. Y yo no necesité escribir todo un sistema operativo libre tampoco. Escribí algunas partes e invité a otras personas a unírseme escribiendo otras partes. Así que establecí un ejemplo a seguir. Y dije: «Yo voy en esta dirección. Únete a mí y llegaremos allí». Y se unió la suficiente cantidad de gente, y allí llegamos. De este modo, si piensas en términos de cómo voy a hacer todo este trabajo gigantesco, puede ser desalentador. Así que la idea es: no lo mires de esa manera. Piensa en términos de dar un paso y comprender que, luego de que diste un paso, otra gente dará más pasos y, juntos, el trabajo será realizado, eventualmente.

Asumiendo que la humanidad no se elimine a sí misma, el trabajo que hacemos hoy al producir una infraestructura educativa libre, las fuentes del libre aprendizaje para el mundo, será útil por tanto tiempo como la humanidad exista. ¿Y qué si lleva veinte años lograr que se haga? No pienses en términos del tamaño del trabajo completo. Piensa en términos de la parte que vas a hacer. Eso le mostrará a la gente que puede hacerse, y entonces otros harán otras partes.

Capítulo 20
Software libre: libertad y cooperación[1]

Introducción

MIKE URETSKY. Soy Mike Uretsky. Estoy en la Escuela Empresarial Stern. También soy uno de los codirectores del Centro de Tecnologías Avanzadas. Os doy la bienvenida en nombre de toda la gente del Departamento de Ciencia Informática. Quería comentar algo antes de pasarle la palabra a Ed que va a presentar al conferenciante.

El papel de la Universidad es el de un lugar donde se fomenta el debate y se tienen interesantes discusiones. Y el papel de una Universidad avanzada es tener discusiones especialmente interesantes. Esta exposición en concreto, este seminario entra dentro de ese modelo. Encuentro particularmente interesante la discusión sobre el *open source*. En cierto sentido… [Risas del público]

RICHARD M. STALLMAN. Yo hago software libre. El *open source* es un movimiento diferente. [Risas del público y aplausos]

MIKE URETSKY. Cuando entré en el sector por primera vez en la década de 1960, el software era esencialmente libre. Hemos pasado por ciclos. Era libre, y después los fabricantes de software, apremiados por la necesidad de expandir sus mercados, lo empujaron hacia otras direcciones. Muchos de los desarrollos que tuvieron lugar con la llegada del PC se desplazaron exactamente con la misma secuencia cíclica.

Un filósofo francés muy interesante, Pierre Levy, habla sobre un desplazamiento en esta dirección y sobre la entrada en el ciberespacio como algo no sólo relacionado con la tecnología sino con la reestructuración social, la reestructuración política, a través de la transformación de los modos de relación que mejorarán el bienestar de la humanidad. Esperamos que este debate sea un nuevo movimiento en esa dirección, que este debate sea algo que atraviese muchas de las disciplinas que normalmente funcionan como compartimentos estancos dentro de la Universidad. Esperamos asistir a una discusión interesante. Ed, cuando quieras...

ED SCHONBERG. Soy Ed Schonberg, del Departamento de Ciencia Informática del Instituto Courant. Permitidme daros la bienvenida a este evento. Los presentadores son

[1]Lo que sigue es una transcripción corregida de la conferencia dictada en la New York University el 29 de mayo de 2001.

normalmente un aspecto particularmente inútil en las exposiciones públicas, pero en este caso, en realidad, han servido para un propósito útil, como Mike ha demostrado fácilmente, dado que, por ejemplo, un presentador puede permitir con sus comentarios inexactos, que el conferenciante ponga orden, corrija y afine considerablemente los parámetros del debate. [Risas del público]

Así, permitidme hacer una presentación lo más breve posible de alguien que no la necesita. Richard es el ejemplo perfecto de alguien que, actuando localmente, empezó a pensar globalmente a raíz de los problemas de inaccesibilidad al código fuente de los drivers de impresora en el laboratorio de inteligencia artificial del MIT, hace muchos años. Ha desarrollado una filosofía coherente que nos ha obligado a todos a replantearnos nuestras ideas sobre cómo se produce software, sobre qué significa la propiedad intelectual y sobre lo que de verdad representa la comunidad del software. Permitidme dar la bienvenida a Richard Stallman. [Aplausos]

Software libre: libertad y cooperación

STALLMAN. ¿Alguien me puede dejar un reloj? [Risas del público] Gracias. Bien, quiero dar mi agradecimiento a Microsoft por darme la oportunidad de estar en este estrado. [Risas del público]. Durante las últimas semanas, me he sentido como un escritor cuyo libro ha sido accidentalmente prohibido en alguna parte.[2] [Risas del público]. Excepto que todos los artículos sobre el mismo están dando el nombre del autor equivocado, porque Microsoft describe la GNU GPL como una licencia *open source* y la mayoría de las coberturas periodísticas han hecho lo mismo. La mayoría de la gente, por supuesto de forma inocente, no se da cuenta de que nuestro trabajo no tiene nada que ver con el *open source*, de que en realidad hicimos la mayor parte de él antes de que la gente ni siquiera acuñara el término *open source*.

Nosotros estamos en el movimiento del software libre y voy a hablar sobre qué es el movimiento del software libre, sobre qué significa, sobre qué hemos hecho, y, dado que esto en parte está patrocinado por una escuela empresarial, diré algunas cosas más de lo que normalmente suelo decir sobre cómo se relaciona el software libre con el ámbito empresarial y con otras áreas de la vida social.

Bien, algunos de vosotros quizás no escriba nunca programas informáticos, pero tal vez sabéis cocinar. Y si sabéis cocinar, a no ser que realmente seáis unos genios, probablemente uséis recetas. Y si usáis recetas, probablemente habréis tenido la experiencia de obtener la copia de una receta a través de un amigo que la comparte. Y probablemente también hayáis tenido la experiencia —a no ser que seáis unos auténticos principiantes— de cambiar la receta. Una receta dice ciertas cosas, pero no tienes qué hacer exactamente lo que dice. Puedes dejar fuera algunos ingredientes. Añadir algunos champiñones, por que te gustan los champiñones. Echarle menos sal porque tu médico te dijo que deberías prescindir de la sal. Lo que sea. Puedes incluso hacer

[2]Poco menos de un mes antes, el vicepresidente de Microsoft, Craig Mundie, dio una charla en la que atacó el software libre —llamándolo «open source».

cambios más grandes, según tus habilidades. Y si has hecho cambios en una receta y se la cocinas a tus amigos, y les gusta, alguno de tus amigos dirá: «Oye, ¿me puedes pasar la receta?». Y entonces, ¿qué haces? Puedes apuntar tu versión modificada de la receta y hacerle una copia a tu amigo. Esto es lo que se hace naturalmente con cualquier tipo de recetas útiles.

Bueno, una receta es muy parecida a un programa informático. Un programa informático se parece mucho a una receta: una sucesión de pasos realizados para conseguir algún resultado deseado. De modo que es igual de natural hacer lo mismo con los programas informáticos. Pásale una copia a tu amigo. Haz cambios en ella porque el trabajo para el que fue escrita no es exactamente el mismo que quieres hacer. Este fue de gran ayuda para otro, pero tu trabajo es diferente. Y, después de cambiarla, probablemente sea útil para más gente. Quizás tengan que hacer un trabajo como el que tú haces, así que preguntan: «Oye, ¿me puedes dejar una copia?». Por supuesto, si eres una buena persona les darás la copia. Esa es la forma de ser de una persona decente.

Imaginad qué pasaría si las recetas estuvieran empaquetadas dentro de cajas negras. No podríais ver qué ingredientes están usando, por no hablar de cambiarlos, imaginad que hacéis una copia para un amigo, te llaman pirata y te intentan meter unos años en la cárcel. Ese mundo generaría una tremenda afrenta a toda la gente que está acostumbrada a compartir recetas. Pero exactamente así es el mundo del software propietario. Un mundo en el que la usual decencia con los otros está prohibida o coartada.

Y bien, ¿por qué me enteré de esto? Me enteré porque, en la década de 1970, tuve la buena suerte de ser parte de una comunidad de programadores que compartían software. Esencialmente esta comunidad podía reconocer sus ancestros en los comienzos de la programación. En la década de 1970, sin embargo, era un poco extraño que hubiera una comunidad que compartiese software. Y, de hecho, era algo así como un caso extremo; en el laboratorio en el que yo trabajaba, todo el sistema operativo era software desarrollado por la gente de nuestra comunidad y compartíamos cualquier parte de él con cualquiera. Cualquiera era bienvenido para entrar y echar un vistazo, llevarse una copia y hacer lo que quisiera. No había notas de copyright en estos programas. La cooperación era nuestro modo de vida. Y estábamos seguros dentro de ese modo de vida. No luchábamos por él. No teníamos que luchar por él. Simplemente vivíamos así. Y, hasta donde nos concernía, habríamos seguido viviendo así. Así que había software libre, pero no un movimiento del software libre.

Sin embargo, más tarde, nuestra comunidad fue destruida por una sucesión de calamidades. Finalmente quedó anulada. Finalmente, el ordenador PDP-10,[3] que usábamos para todo nuestro trabajo, se dejó de fabricar. Nuestro sistema —el *Sistema Incompatible de Uso Compartido*— fue escrito a principios de la década de 1960, de modo que estaba escrito en lenguaje ensamblador. Era con lo que se solían escribir los sistemas operativos en la década de 1960. Por supuesto, el lenguaje ensamblador estaba orientado para una arquitectura informática particular; si esta deja de fabricarse, todo tu trabajo se con-

[3]Procesador de Datos Programados modelo 10, un servidor usado por muchas investigaciones punteras y organizaciones gubernamentales durante la década de 1970.

vierte en polvo —es inútil. Y eso fue lo que nos pasó. Los 20 años de trabajo de nuestra comunidad se convirtieron en polvo.

Pero antes de que pasara esto, tuve una experiencia que me preparó, me ayudó a ver qué hacer, me ayudó a prepararme para ver qué hacer cuando esto sucediera, porque en cierto momento Xerox donó al laboratorio de inteligencia artificial, en el que yo trabajaba, una impresora láser; era un regalo realmente hermoso, ya que era la primera vez que alguien fuera de Xerox tenía una impresora láser. Era muy rápida, imprimía una página por segundo, muy buena en muchos aspectos, pero muy inestable, porque en realidad era una fotocopiadora de oficina de alta velocidad que había sido transformada en impresora. Y como ya sabéis, las fotocopiadoras se atascan, aunque siempre hay alguien para arreglarlas. La impresora se atascaba y nadie lo podía ver. Así que se quedaba atascada mucho tiempo.

Bien, teníamos una idea de cómo tratar este problema. Modificarla de modo que cada vez que la impresora se atascaba, el ordenador que controlaba a la impresora pudiera decir a nuestra máquina de uso compartido y a los usuarios que estaban esperando la salida de impresión que había que arreglar la impresora, para que así al menos supieran que estaba atascada... por supuesto, si estás esperando que se impriman tus trabajos y sabes que la impresora está atascada, no te sientas y esperas una eternidad, sino que te pones a arreglarla.

Sin embargo, llegados a ese punto, estábamos completamente bloqueados, dado que el software que controlaba esa impresora no era software libre. Había venido con la impresora y era simplemente un binario. No podíamos tener el código fuente, Xerox no nos permitía tener el código fuente. Así, a pesar de nuestra habilidad como programadores —después de todo, habíamos escrito nuestro propio sistema de uso compartido— estábamos completamente imposibilitados para añadir esta característica al software de la impresora.

Lo único que podíamos hacer era sufrir con la espera. Podía llevar una hora o dos conseguir que se imprimieran tus trabajos, porque la máquina estaba atascada casi todo el tiempo. Esperabas una hora suponiendo: «Se que va a estar atascada. Esperaré una hora y me pasaré a recoger mi copia». Y entonces veías que había estado atascada todo el tiempo y que en realidad nadie la había arreglado. Así que la arreglabas y volvías a esperar otra media hora. Entonces, volvías y la veías atascada otra vez antes de que hubiera llegado a imprimir tu trabajo. Imprimía durante tres minutos y estaba atascada treinta minutos. Frustración total. Pero lo peor era saber que podíamos haberla arreglado, pero alguien, por su propio egoísmo, nos estaba coartando e impidiendo que mejorásemos el software. Así que, por supuesto, nos sentíamos algo resentidos.

Entonces oí que alguien de la Universidad Carnegie Mellon tenía una copia de ese software. Iba a hacerle una visita allí, así que fui a su oficina y dije: «Buenas, soy del MIT. ¿Podrías dejarme una copia del código fuente de la impresora?». Y él dijo, «no, prometí que no te daría ninguna copia». [Risas del público]. Yo estaba petrificado. De verdad, estaba muy enfadado y no sabía cómo podía expresarlo. Todo lo que me salía era dar media vuelta y salir de la habitación. Quizás di un portazo. [Risas del público]. Y pensé sobre ello más adelante, porque me daba cuenta de que no había visto sólo a un

capullo aislado, sino que se trataba de un fenómeno social importante y que afectaba a mucha gente.

Tuve suerte, sólo me tocó probar un poquito. Otros tenían que vivir con esto todo el tiempo. Así que pensé sobre este asunto en profundidad. Observad, él había prometido que se negaría a cooperar con nosotros —sus colegas del MIT—. Nos había traicionado. Pero no sólo nos lo hizo a nosotros. El caso es que también te lo hizo a ti. [Señala a un miembro del público]. Y creo que muy probablemente también te lo hizo a ti. [Señala a otro miembro del público. El público ríe]. Y probablemente también te lo hizo a ti. [Señala a un tercer miembro del público]. Posiblemente se lo hizo a la mayoría de la gente que está en esta sala —excepto tal vez a unos pocos que aún no habían nacido en 1980.— Puesto que había prometido que se negaría a cooperar con casi toda la población del planeta Tierra. Había firmado un acuerdo de no divulgación.

Bien, este fue mi primer encuentro con un acuerdo de no divulgación y aprendí una lección importante —importante porque la mayoría de los programadores nunca la aprenden. Este era mi primer encuentro con un acuerdo de no divulgación y yo era la víctima. Yo, y todo mi laboratorio, éramos las víctimas. Y la lección que me enseñó es que los acuerdos de no divulgación tienen víctimas. No son inocentes. No son inofensivos. La mayoría de los programadores se encuentran por primera vez con un acuerdo de no divulgación cuando se les ofrece firmar uno. Y siempre hay alguna tentación —si firmas vas a obtener alguna propinilla. Así que se inventan excusas. Dicen, «bueno, no va a conseguir una copia pase lo que pase, así que, ¿por qué no unirme a la conspiración para marginarle?» Dicen, «así es como siempre se ha hecho. ¿Quién soy yo para ponerme en contra?» Dicen, «si no firmo esto, lo hará otro». Excusas varias para acallar sus conciencias.

Pero cuando alguien me ofreció la firma de un acuerdo de no divulgación, mi conciencia ya estaba sensibilizada. Recuerdo cómo me había enfadado cuando alguien prometió que no me ayudaría a mi y a todo mi laboratorio a resolver nuestro problema. No podía darme la vuelta y hacerle exactamente lo mismo a alguien que nunca me había hecho ningún daño. Si alguien me pidiera que prometiera no compartir alguna información útil con un enemigo odiado, yo aceptaría. Si alguien ha hecho algo malo, se lo merece. Pero los desconocidos no me han hecho ningún daño. ¿Cómo podrían merecerse ese tipo de maltrato? No puedes permitirte empezar a tratar mal a todo el mundo. Si no te conviertes en un depredador para la sociedad. Así que dije: «Muchas gracias por ofrecerme este bonito paquete de software. Pero, en las condiciones que ustedes exigen, no puedo aceptarlo con la conciencia tranquila, así que me las arreglaré sin él. Muchísimas gracias». Y de este modo, nunca he firmado a sabiendas un acuerdo de no divulgación para información de utilidad técnica general, como el software.

Ahora bien, hay otros tipos de información que implican diferentes asuntos éticos. Por ejemplo, está la información personal. Si quisieras hablar conmigo sobre lo que estaba pasando entre tú y tu novio, y me pidieras que no se lo contara a nadie, yo estaría de acuerdo en guardarte ese secreto, porque no es información de utilidad técnica general.

Al menos, es probable que no sea útil a todo el mundo. [Risas del público]. Hay

pocas posibilidades —y aún así es una posibilidad— de que alguien quisiera revelarme alguna maravillosa nueva técnica sexual, entonces sentiría el deber moral de hacérselo saber al resto de la humanidad, y así todo el mundo podría beneficiarse de ella. [Risas del público]. Así que tendría que poner una cláusula en esa promesa.

Si sólo son detalles sobre quien quiere qué y quién está enfadado con quién, y ese tipo de culebrones. . . *eso* lo puedo guardar en secreto; pero no puedo retener algo de cuyo conocimiento la humanidad puede beneficiarse tremendamente. Veréis, el propósito de la ciencia y la tecnología es crear información útil para la humanidad, para ayudar a la gente a vivir mejor. Si prometemos que retendremos esa información —si la guardamos en secreto— entonces estaremos traicionando el objetivo de nuestro sector. Y eso, decidí que no debía hacerlo.

Pero mientras tanto mi comunidad se había ido a pique y eso me dejaba en una mala situación. Fijaros, todo el Sistema Incompatible de Uso Compartido estaba obsoleto, porque el PDP-10 estaba obsoleto, de este modo no había forma de que pudiera seguir trabajando como desarrollador de sistemas operativos tal y como lo había estado haciendo. Esta actividad dependía de formar parte de la comunidad, usar el software de la comunidad y mejorarlo. Ya no era posible, lo cual me planteó un dilema moral. ¿Qué iba a hacer? Porque la posibilidad más obvia significaba ponerme en contra de esa decisión que había tomado. La posibilidad más obvia era adaptarme a los cambios del mundo. Aceptar que las cosas eran diferentes y que yo debería abandonar esos principios y empezar a firmar acuerdos de no divulgación para sistemas operativos propietarios, muy probablemente escribir también software propietario. Así, me di cuenta de que podría divertirme escribiendo código y que podría ganar dinero —especialmente si lo hiciera en cualquier parte que no fuera el MIT—, pero al final, hubiera tenido que repasar mi carrera y decir «me he pasado la vida construyendo muros para dividir a la gente», estaría avergonzado de mi vida.

Así que busqué otra alternativa, y había una obvia. Podía dejar el sector del software y dedicarme a otra cosa. Bien, no tengo otras habilidades reseñables, pero estoy seguro de que podría haber llegado a ser camarero. [Risas del público]. No en un restaurante de lujo; no me contratarían, pero podría ser camarero en algún sitio. Y muchos programadores me decían «la gente que contrata programadores exige esto, esto y esto. Y si no hago estas cosas, me moriré de hambre». Es literalmente la expresión que usan. Bien, como camarero no vas a morirte de hambre. [Risas del público]. Así que, realmente, no estás en peligro. Pero —y, sabéis, esto es importante— a veces puedes justificarte por hacer cosas que dañan a los demás diciendo que de otro modo te puede pasar algo peor. Si *de verdad* te fueras a morir de hambre, estaría justificado que escribieras software propietario. [Risas del público]. Si alguien te apunta con una pistola, entonces yo diría que se te puede perdonar. [Risas del público]. Pero yo encontré una forma de sobrevivir sin hacer algo inmoral, así que esa excusa no valía. Me di cuenta, de todos modos, de que ser camarero no iba a resultarme divertido y significaría derrochar mis habilidades como desarrollador de sistemas operativos. Evitaría el mal uso de mis habilidades. Desarrollar software propietario sería un mal uso de mis habilidades. Apoyar que los demás vivan en el mundo del software propietario sería usar mal mis

habilidades. Así que es mejor malgastarlas que hacer una mal uso de ellas, pero aún así no era algo bueno de verdad.

Por estos motivos decidí buscar otra alternativa. ¿Qué puede hacer un desarrollador de sistemas operativos para mejorar realmente la situación y hacer del mundo un lugar mejor? Me di cuenta de que un desarrollador de sistemas operativos era exactamente lo que se necesitaba. El problema, el dilema para mí y para todos los demás era que todos los sistemas operativos disponibles para los ordenadores modernos eran propietarios. Los sistemas operativos libres estaban destinados a los ordenadores viejos y obsoletos, ¿verdad? Así que para los ordenadores modernos, si querías comprar un ordenador moderno y usarlo, estabas obligado a usar un sistema operativo propietario. De modo que si un desarrollador de sistemas operativos escribiera otro sistema operativo y dijera «venid todos a compartir esto; sois bienvenidos», se daría una escapatoria al dilema, otra alternativa para todo el mundo. Así que me di cuenta de que había algo que podía hacer que podría resolver el problema. Tenía justo las habilidades necesarias para poder hacerlo. Y posiblemente era la cosa más útil que yo podía imaginarme en relación a lo que sería capaz de hacer con mi vida. Se trataba de un problema que nadie más estaba intentando resolver. Era algo así como quedarme sentado, ver empeorar las cosas y que no hubiera nadie más que yo. De este modo, sentí: «Soy el elegido. Tengo que trabajar en esto. ¿Si no lo hago yo, quién lo hará?». Decidí que desarrollaría un sistema operativo libre, o moriría en el intento... de viejo, por supuesto. [Risas del público].

Por supuesto, tuve que decidir qué tipo de sistema operativo debería ser. Había que tomar algunas decisiones técnicas de diseño. Decidí hacer que el sistema fuera compatible con Unix por una serie de razones. En primer lugar, acababa de ver como se quedaba obsoleto a un sistema operativo que realmente amaba, porque estaba escrito para un modelo particular de ordenador. No quería que eso sucediera otra vez. Necesitábamos tener un sistema que se pudiera migrar. Bien, Unix era un sistema portable. Así que si seguía el diseño de Unix, tenía bastantes posibilidades de hacer un sistema que también se pudiera migrar y funcionar en otros sistemas. Y además, por qué no hacerlo compatible en todos sus detalles. La idea es que los usuarios odian los cambios que los hacen incompatibles. Si hubiera diseñado el sistema de acuerdo a mi manera favorita —lo cual me hubiera encantado, estoy seguro— habría producido algo incompatible. Los detalles serían diferentes. Así que, si hubiera escrito el sistema, los usuarios me habrían dicho: «Bien, esto es muy bonito, pero es incompatible. Cambiarse costará mucho trabajo. No nos podemos permitir tener tantos problemas sólo para usar tu sistema en lugar de Unix, así que nos quedamos con Unix».

Ahora bien, si de verdad quería crear una comunidad en la que hubiera gente, gente usando este sistema libre y disfrutando de los beneficios de la libertad y la cooperación, tenía que producir un sistema que la gente usara, un sistema cuya migración desde otro sistema fuera fácil, que no tuviera un impedimento que lo hiciera fracasar desde el mismo principio. Bueno, en realidad hacer el sistema compatible con Unix adelantó ya todas las decisiones de diseño inmediatas, porque Unix está formado por muchos elementos, que se comunican por interfaces que están más o menos documentadas. De modo que si quieres compatibilidad con Unix, tienes que sustituir cada elemento, uno

por uno, con un elemento compatible. De este modo, las restantes decisiones de diseño se toman dentro de cada elemento y pueden ser tomadas después por cualquiera que decida escribir ese elemento. No tienen que tomarse desde el principio.

Todo lo que teníamos que hacer para empezar el trabajo era encontrar un nombre para el sistema. Bueno, nosotros los hackers siempre buscamos nombres divertidos o traviesos para los programas, porque pensar que a la gente le haga gracia el nombre es la mitad de la diversión de escribir el programa. [Risas del público]. Teníamos una tradición de acrónimos recurrentes para decir que el programa que estás escribiendo es parecido a algún programa existente. Puedes ponerle de nombre un acrónimo recurrente que diga: esto no es lo otro [this one's not the other]. Así, por ejemplo, había muchos editores de texto Tico en la década de 1960 y 1970, normalmente se les llamaba tal y cual Tico. En ese momento, un hacker espabilado llamó al suyo Tint, lo que quería decir Tint No Es Tico[4] —el primer acrónimo recurrente—. En 1975, creé el primer editor de texto Emacs; había muchas imitaciones de Emacs, muchas de las cuales se llamaban tal y cual Emacs, pero una se llamaba Fine, por *Fine Is Not Emacs*, y también estaba Sine, por *Sine Is Not Emacs*, y Eine por *Eine is not Emacs*, y Mince por *Mince Is Not Complete Emacs*. [Risas del público]. Esa era una imitación descarada. Y entonces Eine fue reescrito casi por completo, y la nueva versión se llamó Zwei por *Zwei Was Eine Initially*.[5] [Risas del público].

Así, busqué un acrónimo recurrente para Algo No Es Unix [Something Is Not Unix]. Y probé con las 26 letras, y descubrí que ninguna de ellas era una palabra. [Risas del público]. Hmm, prueba de otra forma. Hice una contracción. De este modo podría tener un acrónimo de tres letras, para Algo No Unix. Probé con varias letras y encontré la palabra GNU[6] —la palabra GNU es la más graciosa de todo la lengua inglesa—. [Risas del público]. Ahí estaba. Por supuesto, el motivo de que sea gracioso es que según el diccionario, se pronuncia «new». Por eso la gente lo usa en muchos juegos de palabras. Dejadme que os diga que este es el nombre de un animal que vive en África. Y la pronunciación africana tenía un golpe seco. [Risas del público]. Quizás todavía lo tenga. Y así, los colonos europeos, cuando llegaron allí, no se molestaron en aprender a dar ese golpe seco. Así que lo eliminaron y escribieron una «g» que significaba «hay otro sonido que supuestamente va aquí pero nosotros no lo pronunciamos». [Risas del público]. De todos modos, esta noche me voy a Sudáfrica y les he rogado, espero que me encuentren a alguien que me enseñe a pronunciar golpes secos para que sepa pronunciar GNU de la forma correcta cuando se trata del animal. [Risas del público].

Pero, cuando es el nombre de nuestro sistema, la pronunciación correcta es «g-NEW» —con la «g» fuerte—. Si hablas de un «nuevo» sistema operativo puedes confundir bastante a la gente, porque hemos trabajado en él desde hace 17 años, así que ya no es muy nuevo. [Risas del público]. Pero todavía es, y siempre lo será, GNU —no importa cuánta gente lo llame Linux por error—. [Risas del público].

[4]En inglés, Tint Is Not Teco, que corresponde a las siglas Tint. [*N. del E.*]

[5]*Eine* y *Zwei* significan uno y dos respectivamente en alemán.

[6]*Gnu* significa «ñu» en inglés. [*N. del E.*]

Así que, en enero de 1984, dejé mi trabajo en el MIT para empezar a escribir elementos de GNU.[7] Aún así, fueron tan amables como para dejarme usar sus recursos. En ese momento, pensé que escribiríamos todos esos elementos y que haríamos un sistema GNU completo, y en ese momento diríamos «Venid a por él» y la gente empezaría a usarlo. Eso no fue lo que pasó. Los primeros elementos que escribí eran buenos recambios, con menos fallos, de algunos elementos de Unix, pero no eran especialmente emocionantes. Particularmente nadie quería obtenerlos y instalarlos. Pero entonces, en septiembre de 1984, empecé a escribir GNU Emacs, que era mi segunda implementación de Emacs y para principios de 1985 ya estaba funcionando. Podría usarlo para todo mi trabajo de edición, lo cual era un gran alivio, porque no tenía intención de aprender vi, el editor de Unix. [Risas del público]. Así que, hasta ese momento, hice mi trabajo de edición con otro equipo y guardé los archivos en la red, de modo que pudiera probarlos. Pero cuando GNU Emacs funcionó lo suficientemente bien como para que yo lo usara, también otra gente quiso usarlo.

De este modo, tuve que pensar bien los mecanismos de distribución. Por supuesto, puse una copia en el directorio anónimo FTP y eso estaba bien para los que estaban en la red —podían ejecutar un archivo tar,[8] pero incluso entonces, en 1985, muchos programadores no tenían acceso a la red. Me mandaban correos electrónicos diciendo. «¿Cómo puedo conseguir una copia?». Tuve que decidir qué les respondería. Bueno, podría haberles dicho «quiero dedicar mi tiempo a escribir más software de GNU, no a escribir cintas, así que por favor encontrad a un amigo que esté en Internet y quiera bajárselo y grabarlo para vosotros», y estoy seguro de que la gente habría encontrado algunos amigos más tarde o mas temprano. Habrían conseguido sus copias.

Pero yo estaba sin trabajo. En realidad, nunca he tenido un trabajo desde que dejé el MIT en enero de 1984. Así que estaba buscando algún modo de hacer dinero mediante mi trabajo con el software libre y por esta razón empecé un negocio de software libre. Anuncié «mandadme 150 dólares y os enviaré una cinta de Emacs». Y los pedidos empezaron a llegar poco a poco. A mediados del año llegaban con cuentagotas.

Estaba recibiendo entre ocho y diez pedidos por mes. Y, si hubiera sido necesario, habría podido vivir sólo de eso, porque siempre he vivido sobriamente. Básicamente vivo como un estudiante. Y me gusta, porque significa que el dinero no me dicta lo que debo hacer. Puedo hacer lo que creo que es importante para mí. Me liberó para hacer lo que parecía merecer la pena. Así que haced un auténtico esfuerzo para evitar quedar atrapados dentro de todos los caros hábitos de vida del americano típico. Porque si lo hacéis, entonces la gente de dinero os dictará lo que tenéis que hacer con vuestra vida. No seréis capaces de hacer lo que es importante para vosotros.

De este modo, iba bien, pero la gente me solía preguntar: «¿Cómo que es software libre si cuesta 150 dólares?». [Risas del público]. Bueno, el motivo de que preguntaran esto es que estaban confundidos por los múltiples significados de la palabra inglesa «libre». Un significado se refiere al precio y el otro se refiere a la libertad. Cuando hablo

[7]Puedes leer el anuncio original del proyecto GNU en «El Manifiesto GNU».

[8]Un programa de archivo de Unix. Combinado con gzip, forma la alternativa de GNU al formato de compresión no libre ZIP.

de software libre, me refiero a la libertad, no al precio. Así que pensad en «libertad de expresión», no en «barra libre». [Risas del público]. A ver, no habría dedicado tantos años de mi vida a asegurarme de que los programadores tengan menos dinero. Ese no es mi objetivo. Soy un programador y no me resulta problemático ganar dinero. No dedicaré toda mi vida a ganar dinero, pero no me preocupa ganarlo. Por lo tanto, —y en la medida en que la ética es igual para todo el mundo— tampoco estoy en contra de que otro programador gane dinero. No quiero que los precios sean bajos. La cuestión es la libertad. Libertad para todos los que utilizan el software, tanto si esa persona es un programador como si no.

Llegados a este punto debería daros una definición de software libre. Me centraré mejor en algunos detalles concretos, porque decir sólo «creo en la libertad» es vacuo. Hay muchas libertades en las que puedes creer, y están en conflicto entre ellas, así que la auténtica cuestión política es: ¿cuáles son las libertades importantes, las libertades que debemos asegurarnos que tenga todo el mundo?

Bien, ahora daré una respuesta a esa pregunta para ese área particular del uso del software. Un programa es software libre para ti, como usuario particular, si tienes las siguientes libertades:

- La Libertad Cero es la libertad de ejecutar el programa con cualquier propósito, de la forma que quieras.
- La Libertad Uno es la libertad de ayudarte a ti mismo cambiando el programa para que se ajuste a tus necesidades.
- La Libertad Dos es la libertad de ayudar al prójimo distribuyendo copias del programa.
- Y la Libertad Tres es la libertad de ayudar a construir tu comunidad publicando una versión mejorada de modo que los otros puedan beneficiarse de tu trabajo.

Si tienes todas estas libertades, el programa es software libre, para ti -y esto es crucial. Por eso lo expreso de ese modo. Lo explicaré más adelante, cuando hable sobre la licencia GNU GPL, pero ahora voy a explicar qué significa el software libre, que es una cuestión más básica.

La *Libertad Cero* es bastante obvia. Si ni siquiera se te permite utilizar el programa como quieras, es un programa la hostia de restrictivo. Pero como de hecho sucede, la mayoría de los programas te darán al menos la Libertad Cero. Y la Libertad Cero se sigue, legalmente, como consecuencia de la Libertad Uno, Dos y Tres —así es como funciona la legislación de copyright—. Así que las libertades que distinguen al software libre del software corriente son las Libertades Uno, Dos y Tres, por eso hablaré más sobre ellas y de por qué son importantes.

La *Libertad Uno* es la libertad de ayudarte a ti mismo modificando el software para que se ajuste a tus necesidades. Esto puede significar arreglar los fallos. Puede significar añadir nuevas características. Puede significar migrarlo a un sistema informático

distinto. Puede significar traducir todos los mensajes de error al navajo. Deberías ser libre de hacer cada modificación que quieras hacer.

Ahora bien, es obvio que los programadores profesionales pueden hacer uso de esta libertad de forma muy efectiva, pero no solo ellos. Cualquiera con un mínimo de inteligencia puede aprender un poco de programación. Hay trabajos difíciles y hay trabajos fáciles, y la mayoría de la gente no va a aprender lo suficiente como para hacer trabajos difíciles. Pero mucha gente puede aprender lo suficiente para hacer trabajos fáciles, del mismo modo que hace 50 años montones y montones de estadounidenses aprendieron a reparar coches, que es lo que permitió a los EEUU tener un ejército motorizado en la Segunda Guerra Mundial y ganar. Es muy importante tener mucha gente que sepa reparar cosas.

Y si eres una persona sociable y realmente no quieres aprender tecnología en absoluto, posiblemente eso significa que tienes muchos amigos y que eres bueno pillándoles para que te deban favores. [Risas del público]. Posiblemente algunos de ellos sean programadores. Así que puedes pedirle a alguno de tus amigos programadores: «¿Podrías cambiarme esto, por favor? ¿Me puedes añadir esta característica?» De este modo, se puede beneficiar mucha gente.

Ahora bien, si no tienes esta libertad, se produce un daño práctico y material a la sociedad. Esta ausencia de libertad te hace prisionero de tu software. Ya expliqué lo que pasaba con la impresora láser. Para nosotros funcionó mal y no podíamos arreglarla porque éramos prisioneros de nuestro software.

Pero también afecta a la moral de la gente. Si el uso del ordenador es siempre frustrante, y la gente lo usa, sus vidas también serán frustrantes, y si lo están usando en su trabajo, su trabajo será frustrante; odiarán su trabajo. Y ya sabéis, la gente se protege a sí misma de la frustración decidiendo no preocuparse. Así que acabas con gente cuya actitud es, «bueno, hoy me he presentado a trabajar. Eso es todo lo que tengo que hacer. Si no puedo progresar, ese no es mi problema; es el problema del jefe». Y cuando esto ocurre, es perjudicial para estas personas y es malo para la sociedad en su conjunto. Esta es la Libertad Uno, *la libertad de ayudarte a ti mismo.*

La *Libertad Dos* es la libertad de ayudar al prójimo distribuyendo copias del programa. Puesto que, para los seres que pueden pensar y aprender, compartir conocimiento útil es un acto de amistad fundamental, cuando estos seres usan ordenadores, este acto de amistad toma la forma de compartir software. Los amigos comparten cosas entre sí. Los amigos se ayudan entre sí. Esta es la naturaleza de la amistad. Y, en realidad, este espíritu de buena voluntad —el espíritu de ayudar al vecino, de forma voluntaria— es el recurso más importante de la sociedad. Marca la diferencia entre una sociedad en la que se puede vivir y la ley de la selva. Su importancia ha sido reconocida por las principales religiones del mundo durante cientos de años y siempre tratan explícitamente de fomentar esta actitud.

Cuando yo iba a la guardería, los profesores intentaban enseñarnos esta actitud — el espíritu de compartir— forzándonos a hacerlo. Se imaginaban que si lo hacíamos, aprenderíamos. Así que decían, «si traéis dulces al colegio, no podéis quedároslos todos para vosotros, tenéis que compartir algo con los otros niños». Nos enseñaban que la

sociedad estaba organizada para enseñar este espíritu de cooperación. ¿Y por qué se tiene que enseñar todo esto? Porque la gente no es totalmente cooperativa. Esta es una parte de la naturaleza humana, pero hay otros aspectos en la naturaleza humana. Y es que la naturaleza humana está compuesta de muchas partes. Así que, si quieres una sociedad mejor, tienes que trabajar para fomentar este espíritu que induce a compartir. Éste espíritu no llegará a acontecer del todo. Es comprensible. La gente también tiene que preocuparse por sí misma. Pero si de algún modo hacemos que este espíritu se extienda, saldremos mejor parados.

Hoy en día, de acuerdo con el gobierno de EEUU, supuestamente los profesores deben hacer exactamente lo contrario. «Ah, Johnny, te has traído software al colegio. Bueno, no lo compartas. Compartir está mal. Compartir significa que eres un pirata». ¿A qué se refieren cuando dicen «pirata»? Lo que están diciendo es que ayudar al prójimo es el equivalente moral de abordar un barco. [Risas del público].

¿Qué habrían dicho Buda o Jesús sobre esto? Bien, coge a tu líder religioso favorito. No sé, quizá Manson habría dicho algo diferente. [Risas del público]. ¿Quién sabe lo que diría L. Ron Hubbard? Pero. . .

PREGUNTA. [Inaudible].

STALLMAN. Por supuesto, está muerto. Pero no lo admiten. ¿Qué?

PREGUNTA. Los demás también, también están muertos. [Risas del público]. [Inaudible]. Charles Manson también está muerto. [Risas del público]. Están muertos, Jesús está muerto, Buda está muerto. . .

STALLMAN. Sí, es verdad. [Risas del público]. Me imagino que, en ese sentido, L. Ron Hubbard no es peor que los otros. [Risas del público]. De todos modos [inaudible].

PREGUNTA. L. Ron Hubbard[9] usaba software libre —le liberaba de Zanu. [Risas del público].

STALLMAN. De todos modos, creo que este es el motivo más importante por el que el software debería ser libre: no nos podemos permitir contaminar el recurso más importante de la sociedad. Es cierto que no es un recurso físico como el aire limpio o el agua potable. Es un recurso psicosocial, pero por todo lo dicho, es igual de real y marca una tremenda diferencia en nuestras vidas. Las acciones que emprendemos influyen en el pensamiento de los demás. Cuando vamos por ahí diciéndole a la gente «No compartáis con los demás», si nos escucha, habremos causado un efecto en la sociedad, y no bueno. Esta es la Libertad Dos, *la libertad de ayudar al prójimo*.

Ah, por cierto, si no tienes esta libertad, no se produce simplemente un daño a los recursos psicosociales, también se produce un daño práctico y material. Si el programa tiene propietario y el propietario impone unas condiciones que hacen que cada usuario tenga que pagar para poder usarlo, algunos dirán: «No importa, me las arreglaré sin él». Y eso es una pérdida, una pérdida deliberadamente infligida. Y lo interesante del

[9] L. Ron Hubbard fue el fundador de la Iglesia de la Cienciología. [*N. del E.*]

software, por supuesto, es que con menos usuarios no significa que tengas que producir menos. Si menos gente compra coches, puedes fabricar menos coches. Eso es un ahorro. Hay recursos que asignar, o no, para fabricar coches, por lo que puedes decir que es bueno que los coches tengan un precio. Impide que la gente desvíe montones de recursos que se van a desperdiciar en fabricar coches que realmente no son necesarios. Pero si cada coche adicional no consumiera recursos, ahorrar al producir estos coches no supondría ningún bien. Bueno, para los objetos físicos, por supuesto, como los coches, siempre van a hacer falta recursos para fabricar unidades adicionales para cada ejemplar adicional.

Pero esto no es cierto para el software. Todo el mundo puede hacer otra copia. Y hacerlo es casi trivial. No consume recursos, excepto una cantidad pequeñísima de electricidad. Así que no hay de dónde ahorrar, no vamos a asignar mejor ningún recurso poniendo este impedimento financiero en el uso del software. A menudo encuentras a gente que acepta las consecuencias de los razonamientos económicos, basados en premisas que no se corresponden con el software y que intenta transplantarlas desde otros ámbitos de la vida dónde sí se pueden aplicar y dónde las conclusiones extraídas pueden ser válidas. Simplemente toman esas conclusiones y asumen que también son válidas para el software, cuando el argumento no tiene ninguna base en el caso del software. Las premisas no funcionan para este caso. Es muy importante examinar cómo se llega a esa conclusión, y de qué premisas depende, para ver donde podría ser válida. Así que esta es la Libertad Dos, *la libertad de ayudar a tu prójimo*.

La *Libertad Tres* es la libertad de ayudar a construir tu comunidad publicando una versión mejorada del software. La gente me solía decir «si el software es libre, entonces a nadie le pagarán por trabajar en él, así que, ¿por qué iba a trabajar nadie en él?» Bueno, por supuesto, estaban confundiendo los dos significados de *libre*, así que su razonamiento se basaba en una mala interpretación del término. Pero, en cualquier caso, esa era su teoría. Hoy, podemos comparar esa teoría con la evidencia empírica y encontramos que a cientos de personas se les paga para escribir software libre, y alrededor 100.000 lo están haciendo de forma voluntaria. Tenemos mucha gente trabajando en el software libre, y esto por varios motivos diferentes.

Cuando publiqué GNU-Emacs por primera vez —la primera parte del sistema GNU que la gente de verdad quería usar— y cuando empezó a tener usuarios, pasado un tiempo, recibí un mensaje que decía, «creo que he visto un fallo en el código fuente, y aquí esta el remedio». Y recibí otro mensaje, «aquí tienes código para añadir una característica nueva». Y otro remedio para un fallo. Y otra característica nueva. Y otra, y otra, y otra, hasta que me empezaron a llover tan rápido que sólo hacer uso de toda esta ayuda que estaba recibiendo suponía mucho trabajo. Microsoft no tiene estos problemas. [Risas del público]

Finalmente, la gente observó este fenómeno. En la década de 1980 muchos de nosotros pensábamos que tal vez el software libre no sería tan bueno como el software no libre, porque no tendríamos tanto dinero para pagar a la gente. Y por supuesto gente como yo, que valora la libertad y la comunidad, dijo, «bueno, usaremos el software libre de todos modos». Merece la pena hacer un pequeño sacrificio en algunas simples

comodidades técnicas a cambio de tener libertad. Pero lo que la gente empezó a observar, hacia 1990, es que nuestro software era en realidad mejor. Tenía más capacidad, era más fiable, que las alternativas propietarias.

A principios de la década de 1990, alguien encontró la forma de medir científicamente la estabilidad del software. Esto es lo que hizo. Tomó diferentes grupos de programas comparables que hacían los mismos trabajos —exactamente los mismos trabajos— en sistemas diferentes, dado que había ciertos programas básicos tipo Unix. Y los trabajos que hacían eran más o menos la misma cosa —o seguían el tipo POSIX— de modo que todos eran iguales en cuanto a los trabajos que hacían; pero eran mantenidos por gente distinta y escritos de forma separada. El código era distinto. Así que dijeron, vale, cogeremos estos programas y los utilizaremos con datos al azar, y mediremos con qué frecuencia se estropean o fallan. Así que lo midieron y el conjunto de programas más fiable era los programas de GNU. Todas las alternativas comerciales, que eran software propietario, eran menos fiables. Así que lo publicó y se lo contó a todos los desarrolladores. Unos pocos años después hizo el mismo experimento con las versiones más novedosas y obtuvo el mismo resultado. Las versiones GNU eran las más fiables. Fijaros, hay clínicas de cáncer y operaciones 911[10] que usan el sistema GNU, porque es muy fiable, y la fiabilidad es muy importante para ellos.

En cualquier caso, existe aún un grupo de gente que considera este beneficio particular como la principal razón por la cual a los usuarios se les debería permitir hacer esta variedad de cosas y tener esas libertades. Si me habéis estado escuchando, os habréis enterado de que, para hablar a favor del movimiento de software libre, hablo sobre temas éticos y sobre el tipo de sociedad en el que queremos vivir, sobre qué produce una buena sociedad, así como sobre los beneficios prácticos y materiales. Todo esto es muy importante, todo esto constituye el movimiento de software libre.

Ese otro grupo de gente —que es llamado *movimiento open source*— sólo cita los beneficios prácticos. Niegan que esta sea una cuestión de principios. Niegan que la gente tenga derecho a la libertad de compartir con su vecino y de comprobar lo que está haciendo el programa y de cambiarlo si no les gusta. Dicen, de todos modos, que permitirle a la gente hacer estas cosas es algo útil. Así que van a las empresas y les dicen, «podríais hacer más dinero si dejáis que la gente haga esto». De modo que podréis ver que, hasta cierto punto, llevan a la gente en una dirección parecida, pero por motivos totalmente distintos, por razones filosóficas fundamentalmente distintas.

En la cuestión más profunda de todas, la cuestión ética, los dos movimientos no están de acuerdo. En el movimiento del software libre decimos «tienes derecho a estas libertades. La gente no puede impedirte que hagas estas cosas». En el movimiento *open source*, dicen: «Sí, pueden impedírtelo si quieren, pero nosotros intentaremos convencerles de que se dignen a permitiros hacer esto». Bueno, han ayudado, han convencido a cierta cantidad de empresas para publicar numerosos elementos de software como software libre en nuestra comunidad. El movimiento *open source* ha ayudado considerablemente a nuestra comunidad y trabajamos juntos en proyectos prácticos. Pero filosóficamente hay un tremendo desacuerdo.

[10]En muchas zonas de EEUU, el 911 es el número para llamadas de emergencia.

Por desgracia, el movimiento *open source* es el que recibe el apoyo de la mayoría de las empresas y así la mayoría de los artículos sobre nuestro trabajo lo describen como *open source*, y mucha gente piensa inocentemente que todos formamos parte del movimiento *open source*. Por eso estoy mencionando esta distinción. Quiero que seáis conscientes de que el movimiento de software libre, que trajo a la existencia a nuestra comunidad y desarrolló el sistema operativo libre, todavía está aquí —y que representamos esta filosofía ética. Quiero que sepáis esto para que no desinforméis a los demás inconscientemente. Pero también para que podáis pensar sobre como os posicionáis.

A qué movimiento apoyéis es cosa vuestra. Podríais estar de acuerdo con el movimiento del software libre y mis puntos de vista. Podríais estar de acuerdo con el movimiento *open source*. Podríais no estar de acuerdo con ninguno. Vosotros decidís dónde os situáis ante estas cuestiones políticas.

Pero si estáis de acuerdo con el movimiento del software libre —si veis que se trata de una cuestión sobre lo que la gente merece decidir en relación al control y la dirección de sus vidas— entonces espero que digáis que estáis de acuerdo con el movimiento del software libre y una forma en que podéis hacerlo es usando el término software libre y ayudando a que la gente sepa que existimos.

Así que la Libertad Tres es muy importante tanto en lo práctico como en lo psicosocial. Si no tienes esta libertad, se produce un daño material, porque este desarrollo comunitario no sucede y no produciremos software fiable y potente. Pero también produce daños psicosociales, que afectan al espíritu de cooperación científica —la idea de que trabajamos juntos para hacer avanzar el conocimiento humano. Veréis, el progreso de la ciencia depende crucialmente de que la gente sea capaz de trabajar junta. Hoy en día, no obstante, a menudo te encuentras con grupos de científicos actuando como si hubiera una guerra entre bandas de científicos e ingenieros. Sin embargo, si no comparten con los demás, todos están bloqueados.

Así que estas son las tres libertades que distinguen al software libre del software típico. La Libertad Uno es la libertad de ayudarte a ti mismo haciendo cambios que se ajusten a tus propias necesidades. La Libertad Dos es la libertad de ayudar a tus amigos distribuyendo copias. Y la Libertad Tres es la libertad de construir tu comunidad haciendo modificaciones y publicándolos para que los use otra gente. Si tienes todas estas libertades, el programa es software libre para ti. Bien, ¿por qué lo defino así, en términos de un usuario particular? ¿Es software libre para ti? [Señala a un miembro del público] ¿Es software libre para ti? [Señalando a otro miembro del público] ¿Es software libre para ti? [Señalando a otro miembro del público] ¿Sí?

PREGUNTA. ¿Puedes explicar un poco la diferencia entre la Libertad Dos y la Tres? [Inaudible].

STALLMAN. Bueno, ciertamente están relacionadas, porque si no tienes libertad de redistribuir en absoluto, ciertamente no tienes libertad de distribuir una versión modificada, pero son actividades diferentes.

La Libertad Dos es: haces una copia exacta y se la pasas a tus amigos, así que ahora tu amigo puede usarla. O quizá hagas copias exactas y se las vendes a un grupo de gente y entonces pueden usarla.

La Libertad Tres es por la que tú haces mejoras —o por lo menos tú piensas que son mejoras y otros podrían estar de acuerdo contigo. Así que esa es la diferencia. Ah, y por cierto, un punto crucial. La Libertad Uno y la Tres dependen de que tengas acceso al código fuente. Porque cambiar un programa exclusivamente binario es extremadamente difícil [risas del público], incluso cambios triviales como usar cuatro dígitos para la fecha[11] [risas del público], si no tienes la fuente. Así que, por motivos prácticos y forzosos, el acceso al código fuente es una condición previa, una obligación, para el software libre.

Así que, ¿por qué lo defino en términos de software libre «para ti»? El motivo es que a veces un mismo programa puede ser software libre para algunas personas y no libre para otras. Bien, esta puede parecer una situación paradójica, así que permitidme poneros un ejemplo para mostraros qué es lo que pasa. Un gran ejemplo —quizás el mayor que haya existido jamás— de este problema fue el sistema X Window, que fue creado en el MIT y lanzado bajo una licencia que lo hacía software libre. Si tenías la versión del MIT con la licencia del MIT, tenías las Libertades Uno, Dos y Tres. Era software libre para ti. Pero entre aquellos que obtuvieron copias estaban varios fabricantes de ordenadores que distribuían sistemas Unix e hicieron los cambios necesarios en X para instalarlos en sus sistemas. Sabéis, probablemente cambiaron sólo unos pocos miles de líneas entre los cientos de miles de líneas de X. Y entonces lo recopilaron y pusieron los binarios en su sistema Unix y lo distribuyeron bajo el mismo acuerdo de no divulgación que tiene el resto del sistema Unix. Y entonces millones de personas obtuvieron estas copias. Tenían el sistema X Window, pero no tenían ninguna de estas libertades. No era software libre para *ellos*.

Así, la paradoja era que el carácter libre de X dependía de dónde hicieras la medición. Si hacías la medición a partir del grupo de desarrolladores, dirías, «observo todas estas libertades. Es software libre». Si hacías la medición entre los usuarios, dirías, «humm, la mayoría de los usuarios no tiene estas libertades. No es software libre». Bueno, la gente que creó X no consideraba esto un problema, porque su objetivo era simplemente la popularidad —el ego, esencialmente. Querían un gran éxito profesional. Querían sentir, «ah, muchísima gente esta usando nuestro software». Y era verdad. Mucha gente estaba usando su software pero no tenía libertad.

Bien, en el proyecto GNU, si esa misma cosa le hubiera pasado al software de GNU, habría sido un fracaso, porque nuestro objetivo no era simplemente ser populares, nuestro objetivo era dar libertad a la gente y apoyar la cooperación, permitir que la gente coopere. Recordad, nunca obliguéis a nadie a cooperar con otra persona, pero aseguraos de que a todo el mundo le está permitido cooperar, de que todo el mundo tenga libertad para cooperar, si él o ella quieren. Si millones de personas tuvieran instaladas versiones no libres de GNU, eso en absoluto sería un éxito. Todo se habría pervertido hasta no ser nada con respecto al objetivo.

[11]Se refiere al fallo «Y2K»: muchos programas viejos guardaban el año con dos dígitos, por eso no estaba claro si la fecha «00» era 2000 o 1900. Se gastaron millones de dólares para reparar este fallo en miles de sistemas informáticos antes del año 2000.

Así que busqué una forma de evitar que eso sucediera. El método al que llegué se llama copyleft. Se llama copyleft porque es algo así como tomar el copyright y darle la vuelta. [Risas del público]. Legalmente, el copyleft funciona sobre la base del copyright. Usamos la legislación de copyright existente, pero la usamos para conseguir un objetivo muy diferente. Esto es lo que hacemos. Decimos: «Este programa tiene copyright». Y, por supuesto, por defecto, eso significa que está prohibido copiarlo, distribuirlo o modificarlo. Pero entonces decimos: «Estás autorizado a distribuir copias de esto. Estás autorizado a modificarlo. Estás autorizado a distribuir versiones modificadas y versiones ampliadas. Cámbialo como te apetezca».

Pero hay una condición. Y la condición, por supuesto, es el motivo por el que nos metemos en estos líos, para que podamos incluir la condición. La condición dice: «Cuando quiera que distribuyas algo que contenga cualquier elemento de este programa, todo ese programa debe ser distribuido bajo estas condiciones, ni más ni menos». Así que puedes cambiar el programa y distribuir una versión modificada, pero cuando lo haces, la gente que lo obtiene de ti debe recibir las mismas libertades que tú recibiste de nosotros. Y no sólo por las partes que copiaste de nuestro programa, también por las otras partes de ese programa que ellos reciben de ti. La totalidad de ese programa tiene que ser software libre para ellos.

La libertad de cambiar y redistribuir este programa se convierte en un derecho inalienable —un concepto de la Declaración de Independencia. Derechos que nos aseguramos que no te puedan ser sustraídos. La licencia específica que encarna la idea de copyleft es la Licencia Pública General GNU, una licencia controvertida porque realmente tiene fuerza como para rechazar a las personas que podrían ser parásitos de nuestra comunidad.

Hay mucha gente que no aprecia los ideales de libertad. Y estaría encantadísima de tomar el trabajo que hemos hecho y usarlo para tener ventaja en la distribución de un programa no libre y tentar así a la gente para que abandone su libertad. Si dejamos que la gente haga eso, el resultado sería que desarrollaríamos programas libres y constantemente tendríamos que competir con versiones mejoradas de nuestros propios programas. Eso no tiene gracia.

Mucha gente también siente: «Estoy deseando dedicar mi tiempo como voluntario para contribuir a la comunidad, pero ¿por qué debería dedicar mi tiempo como voluntario para mejorar el programa propietario de esa empresa?». Algunas personas ni siquiera pensarán que eso es malo, pero quieren que les paguen si hacen eso. Yo, personalmente, preferiría no hacerlo en absoluto.

Pero ambos grupos de personas —los que, como yo, dicen «no quiero ayudar a que ese programa no libre consiga un punto de apoyo en nuestra comunidad» y los que dicen, «claro, yo trabajaría para ellos, pero entonces que me paguen»— tenemos un buen motivo para usar la licencia pública general de GNU. Porque eso le dice a tal compañía «no puedes coger mi obra y distribuirla sin libertad». Puesto que licencias que no son copyleft, como la licencia de X Window, sí lo permiten.

Así que esta es la gran división entre las dos categorías de software libre en lo que respecta a las licencias. Están los programas que tienen copyleft para que la licencia

defienda la libertad del software para todos los usuarios. Y están los programas sin copyleft para los que están permitidas las versiones no libres. Alguien *puede* coger esos programas y despojarles de libertad. Podrías obtener ese programa en una versión no libre.

Ese problema existe hoy. Todavía hay versiones no libres de X Window que se están usando en nuestros sistemas operativos libres. Incluso hay hardware que no soporta realmente más que una versión no libre de X Window. Y ese es un gran problema en nuestra comunidad. No obstante, no diría que el X Window sea algo malo. Diría que los desarrolladores no lo hicieron de la mejor manera que hubieran podido hacerlo. Pero *sí* que lanzaron mucho software que todos podemos usar.

Hay una gran cantidad de matices debajo de la distinción entre perfecto y malo. Hay muchas gradaciones de lo bueno y lo malo. Tenemos que resistir la tentación de decir, «si no lo hiciste lo mejor posible, entonces no eres bueno». La gente que creó X Window hizo una gran contribución a nuestra comunidad. Pero hay cosas que podrían haber hecho mejor. Podrían haber puesto copyleft a algunas partes del programa e impedido que otros distribuyeran esas versiones que niegan la libertad.

Bien, el hecho de que la licencia pública general de GNU defienda tu libertad, de que use la legislación de copyright para defender tu libertad, es el motivo por el que Microsoft está atacando hoy. Veréis, a Microsoft le gustaría de verdad ser capaz de coger todo el código que nosotros escribimos y ponerlo en programas propietarios... que alguien hiciera algunas mejoras... o simplemente algunos cambios para hacerlos incompatibles. [Risas del público].

Con la potencia de marketing de Microsoft, no necesitan hacer versiones mejores para conseguir que su versión suplante a la nuestra. Simplemente tienen que hacerla diferente e incompatible. Y luego colocarla en el escritorio de todo el mundo. Así que verdaderamente no les gusta la GNU GPL. Porque la GNU GPL no les permite hacer eso. No permite «adoptar y ampliar». Dice, si quieres compartir nuestro código en tus programas, puedes. Pero tú también tienes que compartir de forma parecida. Se nos tiene que permitir compartir los cambios que haces. Así que se trata de una cooperación en dos direcciones, una cooperación auténtica.

Muchas empresas —incluso grandes empresas como IBM y HP— están deseando usar, bajo estos parámetros, nuestro software. IBM y HP aportan considerables mejoras al software de GNU. Y también crean más software libre. Pero Microsoft no quiere hacerlo, y hacen saber que los negocios simplemente no pueden utilizar la GPL. Bueno, si los negocios no incluyen a IBM, HP y Sun, quizás estén en lo cierto. [Risas del público]. Más tarde seguiré con esto.

Debería terminar el relato histórico. Observad, empezamos en 1984 no simplemente para escribir software libre sino para hacer algo mucho más coherente: desarrollar un sistema operativo que era por completo software libre. Eso significaba que teníamos que escribir elemento, tras elemento, tras elemento. Por supuesto, siempre estábamos buscando atajos. El trabajo era tan grande que la gente decía que nunca seríamos capaces de acabarlo. Yo pensaba que había por lo menos una posibilidad, pero obviamente merece la pena buscar un atajo. Así que seguimos buscando. ¿Hay algún programa que

otro haya escrito y que podamos adaptar, que conectarlo aquí y así no tendremos que escribir desde cero? Por ejemplo, el sistema X Window. Es cierto que no tenía copyleft, pero era software libre, así que podríamos usarlo.

Bien, yo hubiera querido poner un sistema de ventanas en GNU desde el primer día. Escribí un par de sistemas de ventanas en el MIT antes de empezar con GNU. Y así, aunque Unix no tenía un sistema de ventanas en 1984, decidí que GNU tendría uno. Pero nunca terminamos escribiendo un sistema de ventanas GNU, porque apareció X. Y yo dije: «¡Bien! Un trabajazo que no tendremos que hacer. Usaremos X.» Dije, vamos a coger X y a ponerlo en el sistema GNU. Y haremos que las demás partes de GNU funcionen con X cuando sea apropiado. Encontramos otros elementos de software que habían sido escritos por otros, como el procesador de textos TeX, y un código de biblioteca de Berkeley. En ese momento, existía el Unix Berkeley, pero no era software libre. Inicialmente, este código de biblioteca era de un grupo diferente de Berkeley, que investigaba sobre el punto flotante. Y así encajamos esas piezas.

En octubre de 1985, fundamos la *Free Software Foundation*. Por favor, observad que el proyecto GNU es anterior. La *Free Software Foundation* llegó casi dos años después del que se anunciara el proyecto GNU. La *Free Software Foundation* es una organización benéfica libre de impuestos que recoge fondos para promover la libertad de compartir y de intercambiar el software. Y en la década de 1980 una de las principales cosas que hacíamos con nuestros fondos fue contratar a gente para que escribiera partes de GNU. Programas esenciales, como shell y la biblioteca C, fueron escritos de este modo, así como partes de otros programas. El programa tar, que es absolutamente esencial, aunque no es nada excitante [risas del público], fue escrito de este modo. Creo que el grep de GNU fue escrito de este modo. Y así nos acercábamos a nuestro objetivo.

Hacia 1991 sólo faltaba un elemento principal, que era el kernel. Ahora, ¿por qué habíamos dejado fuera el kernel? Probablemente porque no importa en qué orden hagas las cosas, al menos técnicamente. De todos modos tienes que hacerlas todas. Y en parte porque tenía esperanzas de que seríamos capaces de encontrar un principio de kernel en otro lugar. Y lo hicimos. Encontramos Mach, que había sido desarrollado en Carnegie Mellon. No era todo el kernel; era la mitad inferior del kernel. Así que tuvimos que escribir la mitad superior; cosas como el sistema de archivos, el código de red y así. Pero instaladas sobre Mach funcionan esencialmente como programas de usuario, lo cual debería hacerlas más fáciles de corregir. Puedes corregirlas al mismo tiempo con un depurador de nivel *real source*. Pensé que de ese modo seríamos capaces de conseguir que estas, las partes superiores del kernel, estuvieran listas en poco tiempo. No sucedió así. Estos procesos no sincrónicos y multisegmentados, que se mandan mensajes entre sí, resultaron ser muy difíciles de corregir. El sistema basado en Mach que estábamos usando para ir tirando tenía un entorno de corrección de fallos terrible, no era fiable. Nos llevó años y años conseguir que el kernel de GNU funcionara.

Pero, afortunadamente, nuestra comunidad no tuvo que esperar por el kernel de GNU porque en 1991, Linus Torvalds desarrolló otro kernel libre, llamado Linux. Siguió el diseño monolítico pasado de moda y resulta que consiguió que funcionara mucho más rápido de lo que nosotros conseguimos que funcionara el nuestro. Así que

quizá ese es uno de los fallos que he cometido: esa decisión de diseño. De todos modos, al principio nosotros no supimos de Linux, porque nunca contactó con nosotros para hablar de ello, aunque conocía el proyecto GNU. Sin embargo, lo anunció a otras personas y en otros sitios de la red. Y así otra gente hizo el trabajo de combinar Linux con el resto del sistema GNU para hacer un sistema operativo libre completo, esencialmente, para hacer la combinación de GNU y Linux. Pero no se dieron cuenta de que eso era lo que estaban haciendo. Veréis, ellos decían, «tenemos un kernel —vamos a mirar por ahí a ver qué otros elementos podemos encontrar para juntarlos con el kernel». Así, miraron por ahí, y mira tú por dónde, todo lo que necesitaban ya estaba disponible. Qué buena suerte, dijeron. [Risas del público]. «Todo está aquí. Podemos encontrar todo lo que necesitamos. Vamos a coger todas estas cosas diferentes, a juntarlas y tener un sistema».

No sabían que la mayoría de lo que encontraron eran elementos del sistema GNU. Así, no se dieron cuenta de que estaban encajando Linux en el hueco del sistema GNU. Pensaron que estaban cogiendo Linux y haciendo un sistema a partir de Linux. Así que lo llamaron sistema Linux.

[Un miembro del público dice: «¿Pero no es tener mejor suerte que encontrar el sistema X Window y Mach?», Stallman responde y continúa]. Cierto. La diferencia es que la gente que desarrolló X y Mach no tenía el objetivo de producir un sistema operativo libre completo. Fuimos los únicos en tenerlo. Y fue nuestro tremendo trabajo el que hizo que el sistema existiera. Realmente hicimos una parte mayor del sistema que cualquier otro proyecto. No es una coincidencia, esa gente escribió partes útiles del sistema, pero no lo hicieron porque quisieran terminar el sistema. Tenían otros motivos.

Bien, la gente que desarrolló X pensó que diseñar un sistema de ventanas a través de la red sería un buen proyecto, y lo era. Resulta que nos ayudó a producir un buen sistema operativo libre. Pero eso no es lo que ellos anhelaban. Ni siquiera lo tenían en mente. Fue un accidente. Un beneficio accidental. Ahora, no estoy diciendo que lo que hicieron estuviera mal. Hicieron un gran proyecto de software libre. Eso es algo bueno. Pero no tenían esa visión fundamental. Donde estaba esa visión era en el proyecto GNU.

Y así, nosotros fuimos los que hicimos todos esas piececitas que no hacía nadie más, porque sabíamos que no tendríamos un sistema operativo completo sin ellas. Y aunque fuera totalmente aburrido y no tuviera ningún romanticismo, como tar o mv[12] [risas del público], nosotros las hicimos. O Id. Ya sabéis que no hay nada muy excitante en ld, pero yo escribí uno. [Risas del público]. Y sí que hice esfuerzo para hacer que ocupara una cantidad mínima de disco I/O, de modo que fuera más rápido y manejara programas más grandes. Me gusta hacer un buen trabajo; me gusta mejorar varias cosas del programa mientras las estoy haciendo. Pero el motivo de que lo hiciera no es que tuviera ideas brillantes para un ld mejor. El motivo de que lo hiciera es que necesitábamos uno que fuera libre. Y no podíamos esperar que cualquier otra persona lo hiciese. Así que teníamos que hacerlo nosotros, o encontrar a alguien que lo hiciese.

Así, aunque llegados a este punto, miles de personas y proyectos han contribuido a este sistema, hay un proyecto que es el motivo de que este sistema exista, y es el

[12] Un programa simple que mueve o renombra los archivos

proyecto GNU. Este es básicamente el sistema GNU, con otras cosas añadidas desde entonces.

La práctica de llamar Linux al sistema ha sido un gran golpe para el proyecto GNU, ya que normalmente no recibimos reconocimiento por lo que hemos hecho. Pienso que Linux, el kernel, es un elemento muy útil del software libre y sólo tengo buenas cosas que decir de él. Bueno, en realidad, puedo encontrar unas pocas cosas malas. [Risas del público]. Pero, básicamente, tengo buenas cosas que decir de él. De todos modos, la práctica de llamar «Linux» al sistema GNU es simplemente un error. Me gustaría pediros que, por favor, hagáis un pequeño esfuerzo para llamar GNU/Linux al sistema y de ese modo ayudar a que consigamos nuestra parte del reconocimiento.

[Una persona en el público grita] «¡Necesitáis una mascota! ¡Un animal disecado!» [Stallman responde]. Tenemos uno. [El miembro del público contesta]. «¿De verdad?» [Stallman responde, provocando carcajadas]. Tenemos un animal, un ñu. O sea que sí, cuando dibujéis un pingüino, dibujad un ñu al lado. Pero vamos a dejar las preguntas para el final. Me quedan más cosas que decir.

Entonces, ¿por qué estoy tan preocupado por esto? ¿Por qué pienso que merece la pena importunaros y tal vez rebajar la opinión que tenéis de mí [risas del público], mencionando el tema del reconocimiento? Cuando lo hago, algunos pueden pensar que es porque quiero alimentar mi ego, ¿verdad? Por supuesto, no os pido que lo llaméis «Stallmanix», ¿verdad? [Risas del público. Aplausos].

Os pido que lo llaméis GNU, porque quiero que el proyecto GNU consiga reconocimiento. Y hay una razón muy específica para ello, que es mucho más importante que el reconocimiento de cualquiera, en o por sí mismo. Veréis, estos días, si miráis a vuestro alrededor, en nuestra comunidad la mayoría de la gente que habla sobre el tema y escribe sobre el tema nunca menciona el GNU y jamás mencionan estos objetivos de libertad —estos ideales políticos e ideológicos—, porque el lugar del que estos proceden es GNU.

Las ideas asociadas a Linux, su filosofía, es muy diferente. Esencialmente *es* la filosofía apolítica de Linus Torvalds. Así, cuando la gente piensa que todo el sistema es Linux, tienden a pensar «Ah, todo lo habrá empezado Linus Torvalds. Su filosofía debe ser la que deberíamos estudiar con cuidado». Y cuando oyen algo sobre la filosofía de GNU, dicen: «Tío, esto es tan idealista, debe ser terriblemente poco práctico. Soy un usuario de Linux, no un usuario de GNU». [Risas del público].

¡Qué ironía! ¡Si lo supieran! Si supieran que el sistema que les gustó, o que, en algunos casos, aman; que el sistema por el que se vuelven locos es nuestra filosofía política idealista hecha realidad. . .

Aún así no tendrían por qué estar de acuerdo con nosotros. Pero al menos verían una razón para tomárselo en serio, para darle una oportunidad. Verían como se relaciona con sus vidas. Si cayeran en la cuenta, «estoy usando el sistema GNU. Esta es la filosofía de GNU. Por esta filosofía existe el sistema que tanto me gusta», al menos lo considerarían con la mente mucho más abierta. No significa que todo el mundo vaya a estar de acuerdo. La gente piensa cosas diferentes. Eso está bien —la gente debería

formarse su propia opinión—, pero quiero que esta filosofía reciba el beneficio del reconocimiento por los resultados que ha logrado.

Si miras a tu alrededor en nuestra comunidad, te encontrarás con que en casi todas partes, las instituciones están llamando Linux al sistema. Los reporteros lo llaman en su mayoría Linux. No es correcto, pero lo hacen. Las compañías que empaquetan el producto en su mayoría lo llaman así. Ah, y la mayoría de esos reporteros, cuando escriben artículos, normalmente no lo consideran como un asunto político o social. Normalmente lo están considerando como una pura cuestión de negocios en relación a qué compañías van a tener más o menos éxito, la cual es una cuestión bastante secundaria para la sociedad. Y, si consideras las compañías que empaquetan el sistema GNU/Linux para el uso de la gente, bueno, la mayoría lo llaman Linux. Y todas le añaden software no libre.

Veréis, la GNU GPL considera que si tomas un fragmento de código, y algo de código de un programa protegido por la GPL, y añades algo más de código para producir un programa más grande, ese programa entero tiene que ser lanzado bajo la GPL. Pero podrías poner otros programas separados en el mismo disco —disco duro o CD—, y estos podrían tener otras licencias. Esto se considera como una simple agregación. Esencialmente, distribuir dos programas a la vez a alguien no es algo sobre lo que tengamos nada que decir. Así que de hecho, no es cierto —a veces me gustaría que fuese cierto— que si una empresa usa un programa protegido por la GPL en un producto, el producto entero tenga que ser software libre. No lo es —no toma ese alcance, esa extensión. Se trata del programa entero. Si hay dos programas separados que se comunican entre sí a distancia —como enviándose mensajes entre sí— entonces, por lo general, están legalmente separados. Así estas empresas, añadiendo software no libre al sistema, le están dando a los usuarios una idea muy mala, filosófica y políticamente. Le están contando a los usuarios que «no usar software libre está bien. Incluso nosotros se lo estamos poniendo como un extra».

Si observas las revistas sobre el uso del sistema GNU/Linux, la mayoría de ellas tienen un título como «Linux esto o lo otro». Así la mayor parte del tiempo llaman Linux al sistema. Y están llenas de anuncios de software no libre que puedes instalar sobre el sistema GNU/Linux. Esos anuncios tienen un mensaje común. Dicen: «El software no libre es bueno para ti. Es tan bueno que incluso podrías pagar para conseguirlo». [Risas del público].

Llaman a estas cosas «paquetes de valor añadido», lo cual significa una afirmación de sus valores. Están diciendo: valora la comodidad práctica, no la libertad. Yo no estoy de acuerdo con esos valores, así que los llamo «paquetes de libertad sustraída». [Risas del público]. Porque si te has instalado un sistema operativo libre, entonces estás viviendo en el mundo libre. Disfrutas de los beneficios de la libertad que tantos años hemos trabajado para darte. Aquellos paquetes te dan la oportunidad de atarte con una cadena.

Ahora bien, si observas las exposiciones comerciales dedicadas al uso del sistema GNU/Linux, todas se llaman a sí mismas «exposiciones de Linux». Y están llenas de casetas que exhiben software no libre, poniendo el sello de aprobación al software no

libre. Así, casi en cualquier sitio que mires de nuestra comunidad, las instituciones están autorizando el software no libre, negando totalmente la idea de libertad por la que se desarrolló GNU. Y el único lugar donde la gente se puede encontrar con la idea de la libertad es en contacto con GNU y en contacto con el software libre, con el concepto de software libre. Por eso os pido: por favor llamad GNU/Linux al sistema. Por favor haced consciente a la gente de dónde vino el sistema y por qué.

Por supuesto, simplemente con decir ese nombre no estarás dando una explicación de la historia. Puedes teclear cuatro caracteres más y escribir GNU/Linux; puedes decir dos sílabas más. Pero GNU/Linux tiene menos sílabas que Windows 2000. [Risas del público]. No les estás contando mucho, pero les estás preparando, de modo que cuando oigan hablar de GNU, y de qué va, verán cómo eso se conecta con ellos y con sus vidas. Y eso, indirectamente, marca una tremenda diferencia. Así que por favor, ayudadnos.

Habréis advertido que Microsoft llamó a la GPL una «licencia *open source*». No quieren que la gente piense en términos de que la cuestión sea la libertad. Encontraréis que invitan a pensar a la gente de forma cerrada, como consumidores y, por supuesto, ni siquiera a pensar de una forma muy racional como consumidores, ya que van a elegir productos de Microsoft. Pero no quieren que la gente piense como ciudadanos. Eso va en su contra. Al menos en contra de su actual modelo de negocio.

Ahora bien, el software libre... bueno, os puedo contar cómo se relaciona el software libre con nuestra sociedad. Un tema secundario que podría ser de interés para algunos de vosotros es cómo se relaciona el software libre con los negocios.

Bien, en realidad, el software libre es tremendamente útil para los negocios. Después de todo, la mayoría de las empresas de los países avanzados usan software. Y sólo una fracción diminuta de ellos desarrolla software.

El software libre es tremendamente ventajoso para cualquier empresa que use software, porque significa que están en condiciones de adquirir control. En esencia, el software libre significa que los usuarios tienen el control de lo que hace el programa. Tanto individualmente, si les importa lo suficiente tenerlo, como colectivamente, cuando les importa lo suficiente tenerlo. Cualquiera que se preocupe lo suficiente puede ejercer alguna influencia. Si no te importa, no lo compras. Entonces usas lo que prefiere otra gente. Pero si te importa, entonces tienes una voz. Con el software propietario, esencialmente no tienes ninguna voz.

Con el software libre, puedes cambiar lo que quieres cambiar. Y no importa que no haya programadores en tu empresa; no importa. Si quisieras mover los muros de tu edificio, no tienes que tener una empresa de carpintería. Simplemente tienes que ser capaz de encontrar un carpintero y decir, «¿Qué me cobrarás por este trabajo? ¿Y cuándo lo tendrás terminado?». Y si no hacen el trabajo, puedes ir y encontrar a otro.

Existe un mercado libre de asistencia técnica. De tal forma que cualquier negocio que se preocupe por la asistencia encontrará una tremenda ventaja en el software libre. Con el software propietario, la asistencia es un monopolio, porque una compañía tiene el código fuente —o quizás un pequeño número de empresas, que pagaron una cantidad gigantesca de dinero, tienen el código fuente, como en el caso del programa de fuente compartida de Microsoft, pero son muy pocas. Así que no hay tienes muchas

posibilidades de asistencia. Y eso significa que a menos que seas un auténtico gigante, no les importas. Tu empresa no es lo suficientemente importante para que ellos se preocupen si pierden tu negocio o lo que pase con él. Una vez que empiezas a usar el programa, se imaginan que estás atrapado y obligado a recibir su asistencia, porque cambiarse a un programa diferente es un trabajo gigantesco. Así que terminas haciendo cosas como pagar por el privilegio de informar de un fallo. [Risas del público]. Y una vez que has pagado, te dicen, «bueno, vale, hemos advertido tu informe de fallos. Y en unos pocos meses, puedes comprar una nueva versión, y puedes comprobar si lo hemos arreglado». [Risas del público].

Los proveedores de asistencia de software libre no pueden salirse con la suya. Tienen que complacer a los clientes. Por supuesto, puedes conseguir mucha buena asistencia gratuita. Cuelgas tu problema en Internet. Quizá recibas una respuesta al día siguiente. Pero esto no se puede garantizar, por supuesto. Si quieres estar seguro, será mejor que llegues a un acuerdo con una empresa y les pagues. Y esta es, por supuesto, una de las maneras en que funcionan los negocios de software libre.

Otra ventaja del software libre para empresas que usan software es la seguridad y la privacidad. Y esto también incumbe al uso individual, aunque lo haya sacado a relucir en el contexto de los negocios. Veréis, cuando un programa es propietario, ni siquiera podríais saber qué hace en realidad.

Podría tener características incluidas deliberadamente que si las conocieras no te gustarían. Por ejemplo, podría tener una puerta trasera para permitir al creador introducirse en tu máquina. Podría fisgar en lo que haces y devolver la información. Esto no es inusual. Algunos programas de Microsoft lo hacían. Pero no sólo incumbe a Microsoft. Hay otros programas propietarios que se inmiscuyen en las actividades del usuario. Y ni siquiera notarías si lo está haciendo. Y, por supuesto, incluso asumiendo que el fabricante sea completamente honesto, todo programador comete errores. Podría haber fallos que afectan a tu seguridad que no son culpa de nadie. Pero el asunto es: si no es con software libre, no puedes encontrarlos. Y no puedes arreglarlos.

Nadie tiene tiempo para revisar la fuente de cada programa que instala. No vas a hacer eso. Pero junto al software libre hay una extensa comunidad y hay gente en esa comunidad que revisa las cosas. Tú te beneficias de sus revisiones, porque si hay un fallo accidental, que seguramente puede haber de vez en cuando en cualquier programa, podrían encontrarlo y arreglarlo. La gente es mucho menos propensa a meter deliberadamente un troyano, o un programa para fisgar, si piensan que les pueden pillar. Los fabricantes de software propietario se imaginan que no les van a pillar. Se saldrán con la suya sin ser detectados. Pero un fabricante de software libre se tiene que imaginar que la gente se fijará en ello y verá que está allí. En nuestra comunidad no nos parece que podamos escaquearnos si a los usuarios les metemos por la fuerza una característica que a ellos no les gusta. Sabemos que si a los usuarios no les gusta, harán una versión modificada que no la tenga. Y entonces todos empezarán a usar esa versión.

De hecho, todos podemos razonar suficientemente, podemos suponer con suficientes pasos de antelación que probablemente no incluyamos tal característica. Después de todo, estás escribiendo un programa libre, quieres que a la gente le guste tu versión; no

quieres incluir algo que la mayoría de la gente va a odiar y que después va a elegir una versión modificada en lugar de la tuya. Así que simplemente te das cuenta de que el usuario es el rey en el mundo del software libre. En el mundo del software propietario, el cliente no es el rey, ya que sólo eres un mero cliente. No tienes ninguna voz sobre el software que usas.

A este respecto, el software libre es un nuevo engranaje para que actúe la democracia. El profesor Lessig,[13] ahora en Stanford, advirtió que el código funciona como una especie de ley. Quienquiera que llegue a escribir el código que casi todo el mundo usa, para cualquier intención o propósito, está escribiendo las leyes que conducen la vida de la gente. Con el software libre, estas leyes se escriben de forma democrática. No la forma clásica de la democracia —no hacemos unas grandes elecciones y decimos «que todo el mundo vote la manera en que debería hacerse esta característica». [Risas del público]. En su lugar decimos, esencialmente, aquellos de vosotros que queráis trabajar en implementar una característica de tal manera, hacedlo. Y los que queráis trabajar en implementar esta característica de esa otra manera, hacedlo. Y, de una manera u otra, se hace. Y así, si mucha gente lo quiere de este modo, se hará de este modo. De esta manera, todo el mundo contribuye a la decisión social simplemente dando pasos en la dirección que quiere seguir.

Personalmente eres libre de dar tantos pasos como quieras dar. Un negocio es libre de encargar trabajos pasos como se consideren útiles. Y después de sumar todos estos aspectos, descubrimos hacia qué dirección va la producción de software. A menudo es muy útil ser capaz de sacar elementos de algún programa existente —presumiblemente elementos grandes, por supuesto— y entonces escribir cierta cantidad de código propio, producir un programa que haga exactamente lo que necesitas, lo cual te hubiera costado muchísimo si hubieras tenido que escribirlo desde cero, si no hubieras podido canibalizar elementos grandes de algún paquete de software existente.

Otra cosa, que resulta del hecho de que el usuario sea el rey, es que tendemos a ser muy buenos en cuanto a compatibilidad y estandarización. ¿Por qué? Porque a los usuarios les gusta. Los usuarios son propensos a rechazar un programa que incluya incompatibilidades arbitrarias. Ahora bien, a veces hay cierto grupo de usuarios que realmente necesita algún tipo de incompatibilidad, y entonces la tienen. Está bien. Pero cuando los usuarios quieren seguir un estándar, nosotros los desarrolladores tenemos que seguirlo, y lo sabemos. Y lo hacemos. Por contraste, si te fijas en los fabricantes de software propietario, a menudo encuentran ventajoso no seguir deliberadamente un estándar y no porque piensen que de esa manera le están dando al usuario una ventaja, sino más bien porque se están imponiendo al usuario, lo están atrapando. Incluso te los encontrarás haciendo cambios en sus formatos de archivo de vez en cuando, sólo para forzar a la gente a obtener la última versión.

Los documentalistas[14] se están encontrando ahora con un problema, a menudo no se puede acceder a los archivos escritos por un ordenador de hace diez años; estaban es-

[13] Lawrence Lessig ha escrito la introducción a este libro.

[14] Muchos documentalistas almacenan y comparten miles de archivos por Internet.

critos en software propietario que ahora en su mayor parte se ha perdido. Si estuvieran escritos en software libre, entonces podrían actualizarse y utilizarse; y esas grabaciones no se habrían perdido, no serían inaccesibles. Incluso se estaban quejando de esto en la *National Public Radio*[15] al citar el software libre como una solución. En efecto, al usar un programa no libre para almacenar tus datos, te estás colgando una soga al cuello.

He hablado sobre cómo afecta el software libre a la mayoría de los negocios. Pero ¿como afecta al particular y estrecho ámbito de los negocios de software? Bueno, la respuesta es que no afecta gran cosa. Y la razón es que el 90 por ciento de la industria del software, por lo que se me ha dicho, se dedica a la fabricación de programas personalizados, software que no está en absoluto pensado para ser publicado. Para el software personalizado, esta cuestión, o la cuestión ética de software libre o propietario, no aparece. Veréis, esta cuestión es, ¿sois libres los usuarios de modificar y redistribuir software? Si solo hay un usuario, y a ese usuario le pertenecen los derechos, no hay problema. Ese usuario es libre de hacer todas esas cosas. Así, en efecto, cualquier programa personalizado que fuera fabricado por una compañía para uso doméstico es software libre, siempre que tenga la sensatez de insistir en obtener el código fuente y todos los derechos.

La cuestión no atañe el software que va en un reloj o en un microondas o en el motor de un coche, porque en esos lugares no te descargas software para instalar. No es un auténtico ordenador, en lo que concierne al usuario, así que no implica suficientemente a estas cuestiones como para ser éticamente importantes. Así, en su mayor parte, la industria del software continuará como lo ha hecho hasta ahora. Y lo interesante es que dado que una fracción tan grande de los trabajos está en esa parte de la industria, incluso si no hubiera posibilidades para los negocios de software libre, los fabricantes de software libre podrían conseguir trabajo escribiendo software personalizado. [Risas del público]. Hay muchos; la proporción es muy grande.

Pero, como de hecho sucede, hay negocios de software libre. Hay empresas de software libre, y en la rueda de prensa que voy a dar, se me van a unir personas procedentes de un par de estas empresas. Por supuesto, también hay empresas que no son negocios de software libre pero que desarrollan elementos útiles de software libre, y el software libre que producen es considerable.

Bien, ¿cómo funcionan los negocios de software libre? Bueno, algunos de ellos venden copias. Tú eres libre de copiarlo pero aún así se pueden vender miles de copias al mes. Otros venden asistencia y distintas clases de servicios. Yo, personalmente, durante la segunda mitad de los ochenta, vendía servicios de asistencia técnica de software libre. Básicamente, decía, por 200 dólares la hora cambiaré lo que quieras que cambie en el software GNU que he escrito. Sí, era un precio excesivo, pero al ser un programa del que yo era autor, la gente podía suponer que tendría terminado el trabajo en muchas menos horas. [Risas del público]. Y me ganaba la vida de este modo. En realidad, gana-

[15]National Public Radio es una organización privada sin ánimo de lucro que tiene, a la fecha de esta conferencia, 620 estaciones de radio que emiten noticias y música a diario.

ba más de lo que nunca había ganado antes. También impartí clases. Y seguí haciéndolo hasta 1990, cuando recibí un gran premio[16] y no tuve que hacerlo más.

Sin embargo, fue en 1990 cuando se formó la primera empresa de software libre, Cygnus Support. Su negocio consistía en hacer, básicamente, las mismas cosas que yo había estado haciendo. Ciertamente podría haber trabajado para ellos, si lo hubiera necesitado. Como no lo necesitaba, me pareció que era bueno para el movimiento que me mantuviera independiente de cualquier empresa. De ese modo, podría decir cosas buenas y malas sobre las diferentes empresas de software libre y software no libre, sin entrar en un conflicto de intereses. Me pareció que podría servir más al movimiento. Pero si lo hubiera necesitado para ganarme la vida, claro que habría trabajado para ellos. Es un negocio ético en el que se puede estar. No había motivos de sentirme avergonzado si hubiera trabajado para ellos. Y esa compañía dio beneficios desde el primer año. Se formó con muy poco capital, sólo el dinero que tenían sus tres fundadores. Y siguió creciendo cada año y siendo rentable cada año hasta que se volvieron ambiciosos y buscaron inversores externos y entonces lo desbarataron todo. Pero fueron varios años de éxito, antes de que se volvieron ambiciosos.

Esto ilustra una de las cosas excitantes que tiene el software libre. El software libre demuestra que no necesitas captar capital para desarrollar software libre. Quiero decir, es útil, tenerlo puede ayudar. Si consigues capital, puedes contratar gente y hacer que escriban un montón de software. Pero puedes hacer mucho con una pequeña cantidad de gente. En realidad, la tremenda eficacia del proceso de desarrollar software libre es uno de los motivos de que sea importante para el mundo cambiarse al software libre. También desmiente lo que dice Microsoft, cuando dicen que la GNU GPL es mala porque hace que sea más difícil para ellos captar capital para fabricar software no libre, coger nuestro software libre y poner nuestro código en los programas que no compartirán con nosotros. En esencia, no necesitamos que ellos consigan el capital de ese modo. Haremos el trabajo de todos modos. Ya lo estamos haciendo.

La gente nos solía decir que nunca podríamos hacer un sistema operativo completo. Pues bien, hemos hecho eso y una cantidad de cosas tremendamente mayor. Y yo diría que estamos cerca de desarrollar todo el software publicado de interés general necesario en el mundo. Y esto en un mundo donde más del 90 por ciento de los usuarios todavía no usa software libre, y también en un mundo donde más de la mitad de los servidores web en el mundo funcionan con GNU/Linux con Apache como servidor web.

PREGUNTA. [inaudible] ¿Qué dijiste antes, Linux?

STALLMAN. Dije GNU/Linux.

PREGUNTA. ¿De verdad?

STALLMAN. Sí, si estoy hablando del kernel, lo llamo Linux. Es que es su nombre. El kernel fue escrito por Linus Torvalds, y sólo deberíamos llamarlo por el nombre que escogió, por respeto al autor.

---

[16]El «gran premio» al que se refiere es el MacArthur Fellowship, también conocido por algunos como la «beca de los genios». Es una beca de cinco años otorgada a individuos que muestran un mérito excepcional y prometen un trabajo creativo continuo y en ascenso.

De todas formas, en general, en las empresas la mayoría de los usuarios no usan GNU/Linux. La mayoría de los usuarios domésticos no usan nuestro sistema todavía. Cuando lo hagan, deberíamos automáticamente tener 10 veces más voluntarios y 10 veces más clientes para los negocios de software libre de los que hay. Eso nos llevará a esa magnitud. Así que en este punto, estoy bastante seguro de que podemos hacer el trabajo.

Esto es importante, porque Microsoft apela a que estemos desesperados. Dicen: «La única manera en que puedes tener software para instalar, la única manera de que tengas innovación, es si nos das poder. Permítenos que te dominemos. Déjanos controlar lo que puedes hacer con el software que estás utilizando, de modo que podamos exprimirte mucho dinero y usar una parte de él para desarrollar software, y quedarnos el resto como beneficios».

Bueno, jamás deberías sentirte tan desesperado. Jamás deberías sentirte tan desesperado como para abandonar tu libertad. Eso es muy peligroso.

Otra cosa que Microsoft, bueno, no sólo Microsoft... la gente que no apoya el software libre, es que generalmente adopta un sistema de valores en el que lo único que importa son los beneficios a corto plazo: ¿Cuánto dinero voy a ganar este año? ¿Cuánto trabajo puedo terminar hoy? Pensamiento a corto plazo y pensamiento estrecho. Asumen que es ridículo imaginar que cualquiera pueda nunca hacer un sacrificio en nombre de la libertad.

Ayer,[17] mucha gente estaba haciendo discursos sobre estadounidenses que hicieron sacrificios por la libertad de sus compatriotas. Algunos de ellos hicieron grandes sacrificios. Incluso sacrificaron sus vidas por las libertades de las que todo el mundo en nuestro país ha oído hablar. (Al menos en algunos casos; supongo que tenemos que ignorar la guerra en Vietnam.)

Pero, afortunadamente, mantener nuestra libertad para usar software no exige grandes sacrificios. Sólo son necesarios diminutos, pequeños sacrificios, como aprender una interfaz de línea de comandos si todavía no tenemos una interfaz gráfica de usuario. Como hacer el trabajo de esa manera, porque no tenemos un paquete de software libre para hacerlo de aquella otra manera todavía. Como pagar algo de dinero a una compañía que va a desarrollar tal paquete de software libre, de modo que puedas tenerlo en unos pocos años. Pequeños sacrificios que todos podemos hacer. Y a largo plazo, incluso nosotros nos beneficiaremos de ellos. Sabéis, en realidad es más una inversión que un sacrificio. Sólo debemos tener suficiente visión a largo plazo para darnos cuenta de que es bueno para nosotros que invirtamos en mejorar nuestra sociedad, sin contar cada céntimo de calderilla de quién consigue qué parte del beneficio en esa inversión.

Así que, llegados hasta este punto, puedo decir que básicamente he terminado.

Me gustaría mencionar que hay un nuevo acercamiento a los negocios de software libre propuesto por Tony Stanco, que él llama «desarrolladores libres», que implica una estructura empresarial que espera distribuir eventualmente cierta cantidad de beneficios a todos los autores de software libre que se hayan unido a la organización. Ahora

[17]El día anterior era el Memorial Day, fiesta nacional de EEUU en la que se conmemora a los héroes de guerra.

están fijándose en la perspectiva de conseguir un gran contrato gubernamental de desarrollo informático en la India, ya que van a usar software libre como base y de ese modo tener tremendos ahorros en los costes.

Bueno, supongo que ahora tengo que pediros que preguntéis.

Turno de preguntas

PREGUNTA. ¿Cómo podría una compañía como Microsoft incluir un contrato de software libre?

STALLMAN. Bueno, realmente, Microsoft está planeando desplazar muchas de sus actividades a los servicios [de asistencia]. Y lo que planean hacer es algo sucio y peligroso, que es atar los servicios a los programas, un programa un servicio, en una suerte de zigzag. De modo que para usar un servicio, tienes que estar usando este programa de Microsoft, esto va a significar que necesitas usar este servicio, este programa de Microsoft... así que todo está atado. Ese es su plan.

Ahora bien, lo interesante es que vender esos servicios no implica ninguna cuestión ética sobre la elección entre software libre o no libre. Perfectamente, podría ser que ellos tuvieran ese negocio para vender esos servicios en la red. De todos modos, lo que Microsoft planea es usarlos para echar un cerrojazo aún mayor, un monopolio aún más grande sobre el software y los servicios; esto ha sido descrito en un artículo reciente, creo que apareció en el Business Week. Otros dicen que está convirtiendo la red en la Ciudad Empresarial de Microsoft.

Esto es relevante porque el tribunal que cursó la demanda antimonopolio contra Microsoft recomendó dividir la compañía —pero de una forma que no tenía sentido y que no haría ningún bien en absoluto— en dos partes: la de sistemas operativos y la de aplicaciones.

Sin embargo leyendo ese artículo, veo útil y efectivo dividir Microsoft en dos partes, la de servicios y la de software, para obligarles a que se traten sólo a distancia, de modo que la de servicios deba publicar sus interfaces para que cualquiera pueda escribir a un terminal para contactar con ellos, supongo que a cambio de un precio. En cualquier caso, ese es un asunto completamente distinto.

Si Microsoft se divide de esta forma... servicios y software, no serán capaces de usar su software para machacar a la competencia de los servicios de Microsoft. Y no serán capaces de usar sus servicios para machacar a la competencia del software de Microsoft. Nosotros seremos capaces de producir software libre y quizá vosotros lo utilicéis para tratar con Microsoft Servicios, a nosotros no nos importará.

Porque, después de todo, aunque Microsoft es la empresa de software propietario que ha sojuzgado a un mayor número de gente, las otras también han sojuzgado, aunque menos, pero no porque no lo hayan intentado; [risas del público], simplemente no han tenido éxito en sojuzgar a tanta gente. Así, el problema no es Microsoft y sólo Microsoft. Microsoft sólo es el mayor ejemplo del problema que intentamos resolver, que es la usurpación de la libertad de los usuarios para cooperar y formar una sociedad

ética por parte del software propietario. De esta forma, no debemos centrarnos demasiado en Microsoft, aunque me hayan dado la oportunidad de estar en este estrado. Eso no les hace tan importantes. No son el principio y el final de todo.

PREGUNTA. Antes, estabas tratando las diferencias filosóficas entre el software *open source* y el software libre. ¿Qué te parece la actual tendencia de las distribuciones de GNU/Linux que se dirige a soportar sólo plataformas de Intel?

STALLMAN. No veo que aquí haya ninguna cuestión ética. Aunque, de hecho, las empresas fabricantes de ordenadores a veces llevan instalado el sistema GNU/Linux. Aparentemente HP lo ha hecho recientemente. Y no se molestaron en pagar una migración a Windows, porque les habría costado demasiado. Pero conseguir que GNU/Linux fuera instalable supuso, creo, el trabajo cinco ingenieros durante unos cuantos meses. Era fácil de hacer.

Ahora bien, por supuesto, yo animo a la gente a usar autoconf, que es un paquete de GNU que hace más fácil migrar tus programas. Les animo a hacerlo. Cuando una arregla un fallo que no se compilaba en una versión del sistema y te lo manda, deberías incluirlo. Pero no veo que esto sea una cuestión ética.

PREGUNTA. Dos comentarios. Uno es: recientemente, diste una charla en el MIT. Leí la trascripción. Alguien te preguntó sobre las patentes, y tu dijiste que «las patentes son un asunto completamente distinto. No tengo nada que decir sobre eso».

STALLMAN. . Cierto. En realidad tengo mucho que decir sobre las patentes, pero nos llevaría una hora. [Risas del público].

PREGUNTA. Me refería a que, me parece que hay todo un problema detrás. Quiero decir, un motivo por el que las empresas llaman a las patentes y al copyright con palabras como propiedad, pura y dura, es porque intentan construir este concepto para usar el poder del Estado y crear una corriente de monopolio que les sea favorable. Y así, lo que estas cosas tienen en común no es que giren en torno a la misma cuestión, sino que su motivación no es realmente la de servicio público, sino la determinación de las empresas de conseguir un monopolio favorable a sus intereses privados.

STALLMAN. Tienes razón, eso es lo que quieren. Pero hay otro motivo por el que quieren usar el concepto de propiedad intelectual. Y es que no quieren animar a la gente a pensar con cuidado sobre la cuestión de las patentes y del copyright. Porque la legislación de copyright y la de patentes son totalmente distintas, y los efectos del copyright sobre el software y las patentes de software son totalmente diferentes.

Las patentes de software suponen una restricción para los programadores, ya que les prohíbe escribir cierto tipo de programas, mientras que el copyright no hace eso. Con el copyright, tienes permiso de distribución, al menos si los has escrito tú. Así que es tremendamente importante separar los dos problemas.

Tienen un poco en común, sólo en un nivel muy básico, todo lo demás es diferente. Así que, por favor, para fomentar la claridad de las ideas, debatid sobre el copyright o debatid sobre las patentes. Pero no debatáis sobre la propiedad intelectual. Yo tengo opiniones diferentes sobre el copyright, sobre las patentes y sobre el software.

PREGUNTA. Has mencionado al principio que un lenguaje funcional, como las recetas, es como un programa informático. Pero hay mucha diferencia entre las recetas de cocina y los programas informáticos, entre la lengua inglesa y los programas informáticos —la definición de lenguaje funcional es muy amplia. Esto está causando problemas en el caso de los DeCSS y el DVD.

STALLMAN. Para las cosas que no son funcionales por naturaleza, los problemas son parcialmente similares pero también parcialmente distintas. Parte del problema se puede trasladar, pero no todo. Por desgracia, esto implicaría otra hora de charla. No tengo tiempo para entrar en este tema. Pero yo diría que todas las obras funcionales deberían ser libres en el mismo sentido que el software. Ya sabéis, libros de texto, manuales, diccionarios, recetas, y así.

PREGUNTA. Me estaba preguntando sobre la música en Internet. Hay similitudes y diferencias.

STALLMAN. Cierto. Yo diría que la libertad mínima que deberíamos tener para cualquier tipo de información publicada es la libertad de redistribuirla no comercialmente, en toda su extensión. Para obras funcionales, necesitamos la libertad de publicar comercialmente versiones modificadas, porque eso es tremendamente útil para la sociedad. Para las obras no funcionales —ya sabéis, el entretenimiento, o las obras estéticas, o aquellas que manifiestan el punto de vista de una determinada persona— quizá no deberían ser modificadas. Y quizá eso signifique que está bien que el copyright cubra toda su distribución comercial.

Por favor recordad que según la Constitución de EEUU, el propósito de cualquier copyright es beneficiar al público. Se trata de modificar el comportamiento de ciertas sujetos privados, de tal forma que publiquen más libros. Y el beneficio que se deriva es que la sociedad puede discutir diversas cuestiones y aprender. Ya sabéis, tenemos literatura. Tenemos obras científicas. El propósito es fomentarlas. El copyright no existe para el beneficio de los autores, ni mucho menos para beneficio de los editores. Existe para el beneficio de los lectores y de todos aquellos que se benefician de la transmisión de información que se produce cuando la gente escribe y los demás leen. Yo estoy de acuerdo con esa meta.

Sin embargo, en la época de las redes informáticas, el método ya no se sostiene, en la medida en que requiere leyes draconianas que invaden la intimidad y aterrorizan a todo el mundo. Ya sabéis, podéis pasar años de prisión por compartir con el vecino. Esto no pasaba en la época de la imprenta. Entonces el copyright era una regulación industrial. Restringía a los editores. Ahora es una restricción impuesta por los editores al público. Así que las relaciones de fuerza han virado 180 grados, aunque se trate de la misma ley.

PREGUNTA. ¿De este modo, puede ser como cuando se saca música a partir de otra música?

STALLMAN. Cierto. Esa es una interesante…

PREGUNTA. Y las obras nuevas, originales, sabes, también suponen mucha cooperación.

STALLMAN. Sí. Y pienso que probablemente requiera algún tipo de concepto de uso razonable. Ciertamente coger unos pocos segundos de *sample* y usarlo para producir una pieza musical, obviamente eso debería entenderse como uso razonable. Incluso la idea estándar de uso razonable incluye eso, si lo piensas. No estoy seguro de que los tribunales estuviesen de acuerdo, pero deberían. No supondría un auténtico cambio en el sistema tal y como ha existido.

PREGUNTA. ¿Qué piensas sobre la publicación de información pública en formatos propietarios?

STALLMAN. Ah, no se debería hacer. Quiero decir, el gobierno nunca debería obligar a los ciudadanos a usar un programa no libre para acceder, para comunicarse con el gobierno de cualquier manera, en cualquier dirección.

PREGUNTA. Yo he sido, lo que ahora llamaré un usuario de GNU/Linux. . .

STALLMAN. Gracias. [Risas del público].

PREGUNTA. . . . desde los últimos cuatro años, lo único que ha sido problemático para mí, y es algo que es esencial, pienso, para todos nosotros, es navegar por Internet.

STALLMAN. Sí.

PREGUNTA. Algo que decididamente ha sido el punto débil de usar el sistema GNU/Linux ha sido navegar por Internet, porque la herramienta más usada para esto, es Netscape. . .

STALLMAN. No es software libre. Permíteme que te responda a esto. Quiero tratar este punto en aras de comprender mejor. Ha existido una tendencia terrible a que la gente use el navegador Netscape en sus sistemas GNU/Linux. De hecho, todos los sistemas empaquetados comercialmente vienen con él. Así que esta es una situación irónica: trabajamos muy duro para hacer un sistema operativo libre, y ahora, si vas a la tienda, puedes encontrar versiones de GNU/Linux, la mayoría se llaman Linux, y no son libres. Bueno, una parte sí lo es. Pero luego está el navegador Netscape y quizás también otros programas no libres. Así que actualmente es muy difícil encontrar un sistema libre, a no ser que sepas muy bien lo que haces. O, por supuesto, que no te puedas instalar el navegador Netscape.

Ahora bien, en realidad, existen navegadores web libres desde hace muchos años. Existe un navegador web libre que yo solía usar llamado Lynx. Es un navegador web libre que no es gráfico; sólo texto. Esto tiene un tremenda ventaja, y es que no ves los anuncios. [Risas del público]. [Aplausos].

Pero de todos modos, hay un proyecto gráfico libre llamado Mozilla, que ahora está llegando al punto de que pueda ser usado. Yo lo uso ocasionalmente.

PREGUNTA. Konqueror 2.01 es muy bueno.

STALLMAN. Ah, vale. Pues ese es otro navegador gráfico libre. Así, finalmente estamos resolviendo el problema, supongo.

PREGUNTA. ¿Me puedes decir algo sobre esa división filosófico/ética entre el software libre y *open source*? ¿Crees que son irreconciliables?... [Se cambia la cinta de grabación y se pierde el final de la pregunta y el comienzo de la respuesta].

STALLMAN. … a una libertad y a la ética. O si dices, bueno, espero que las empresas decidáis que es más rentable que nos permitáis hacer estas cosas.

Pero, como he dicho, en buena parte del trabajo práctico no importa de verdad cuáles sean las motivaciones políticas de una persona. Cuando una persona se ofrece a ayudar al proyecto GNU, no le decimos: «Tienes que estar de acuerdo con nuestras políticas». Decimos que en un paquete GNU tiene que llamar al sistema GNU/Linux y que tiene que llamarlo software libre. Lo que digas cuando no estés hablando del proyecto GNU, es cosa tuya.

PREGUNTA. La empresa IBM inició una campaña dirigida a las agencias gubernamentales para vender sus nuevas grandes máquinas, en la que usaban Linux como reclamo, y lo llamaban Linux.

STALLMAN. Sí. Por supuesto, realmente es el sistema GNU/Linux. [Risas del público].

PREGUNTA. ¡Eso es! Bueno, pues el director de ventas no sabía nada de GNU.

STALLMAN. Sí. El problema es que ya han decidido con mucho cuidado qué es lo que quieren decir en razón de su provecho. Y la cuestión sobre qué forma de describirlo es más precisa, o justa, o correcta no es la cuestión fundamental que preocupa a una empresa como esa. Bien, en algunas pequeñas empresas, sí, habrá un jefe. Y si el jefe está inclinado a pensar sobre ese tipo de cosas, podría tomar una decisión en ese sentido. Pero no una corporación gigantesca. Es una pena.

Hay otro asunto más importante y más sustancial acerca de lo que está haciendo IBM. Están diciendo que están invirtiendo un billón de dólares en «Linux». Pero quizás también debería ponerle comillas a «invirtiendo en», porque parte de ese dinero está pagando a gente para que desarrolle software libre. Esa es de verdad una contribución a nuestra comunidad. Sin embargo, otras divisiones están pagando a la gente para que escriba software propietario, o para instalar software propietario sobre GNU/Linux y esa no es una contribución a nuestra comunidad. IBM lo está mezclando todo. Algo de esto pueda ser quizás propaganda, y es parcialmente una contribución, aunque en parte sea incorrecta. Es un tema difícil. Parte de lo que hacen es una contribución y parte no, y parte es algo así... pero no exactamente. Y no puedes mezclarlo todo y pensar «¡Guau! Un billón de dólares de IBM». [Risas del público]. Es simplificarlo demasiado.

PREGUNTA. ¿Puedes hablar un poco más sobre las ideas que sustentaron la GPL?

STALLMAN. ¿El pensamiento que sustentó la GPL? En parte quería proteger la libertad de la comunidad contra el fenómeno que ya he descrito con X Window, que también ha sucedido con otros programas de software libre. En realidad, cuando yo empecé a pensar sobre este asunto, el caso de X Window todavía no se había presentado. Pero ya había visto como pasaba lo mismo con otros programas libres. Por ejemplo, TeX. Quería asegurarme de que todos los usuarios tuvieran libertad. De otro modo, me di

cuenta de que podría escribir un programa y quizá mucha gente usaría el programa, pero no tendría libertad. ¿Y qué sentido tendría eso?

Pero la otra cuestión sobre la que estaba pensando era que quería dar a la comunidad la sensación de que no era un felpudo, la sensación de que no tenía que ser presa de cualquier parásito que pasara por ahí. Si no usas copyleft, esencialmente estás diciendo: [hablando dócilmente] «Coge mi código. Haz lo que quieras. No digo que no». Así que cualquiera puede aparecer y decir: [hablando con mucha firmeza] «Ah, quiero hacer una versión no libre de esto. Me lo llevo». Y entonces, por supuesto, posiblemente hagan algunas mejoras, esas versiones no libres podrían atraer a los usuarios y sustituir las versiones libres. Y entonces, ¿qué has conseguido? Simplemente habrás hecho una donación a algún proyecto de software propietario.

Cuando la gente ve que eso está sucediendo, cuando la gente ve que otra gente se lleva lo que hago y jamás dan nada a cambio, puede llegar a ser desmoralizante. Y esto no es sólo una especulación. Lo he visto pasar. Esto era en parte lo que sucedió cuando se extinguió la vieja comunidad a la que yo había pertenecido en la década de 1970. Algunos empezaron a dejar de ser cooperativos. Nosotros asumimos que se estaban lucrando. Ciertamente actuaban como si pensaran que se estaban lucrando. Y nosotros nos dimos cuenta de que ellos podían aprovechar nuestra cooperación y no dar nada a cambio. No había nada que pudiéramos hacer al respecto. Era descorazonador. Nosotros, aquellos de nosotros a los que no nos gustaba esa tendencia, llegamos a discutir sobre el asunto pero no pudimos sacar ninguna idea de cómo detenerlo.

La GPL está diseñada para impedir eso. Dice: «Sí, sois bienvenidos de uniros a la comunidad y usar este código». Podéis usarlo para hacer todo tipo de trabajos. Pero, si publicáis una versión modificada, tenéis que publicarla en nuestra comunidad, como parte de nuestra comunidad, como parte del mundo libre.

Así que, de hecho, hay muchos modos de que la gente se beneficie de nuestro trabajo sin contribuir, como el hecho de que no tenga que escribir ningún software. Mucha gente usa GNU/Linux y no escribe nada de software. No hay obligación de que tengas que hacer algo por nosotros. Pero si haces cierto tipo de cosas, tienes que contribuir. Esto significa que nuestra comunidad no es un felpudo. Y creo que esto dio a la gente la fuerza para sentir que no seríamos pisoteados por todo el mundo. Nos alzaremos por esta causa.

PREGUNTA. Teniendo en cuenta el software libre sin copyleft, dado que cualquiera puede pillarlo y hacerlo propietario, ¿no es posible que alguien lo pille, haga algunos cambios y lance todo bajo la GPL?

STALLMAN. Sí, es posible.

PREGUNTA. Entonces eso convertiría todas las copias futuras en GPL.

STALLMAN. A partir de esa rama. Esto es una de las razones por lo que no lo hacemos. Me voy a explicar. Podríamos, si quisiéramos, coger el X Window y hacer una copia protegida por la GPL y hacer cambios en ella. Pero hay todavía un grupo más grande de gente trabajando en mejorar X Window sin la GPL. Así que, si hiciéramos eso, nos

estaríamos aprovechando de ellos. Y eso no es una forma muy amable de tratarlos. Ellos son parte de nuestra comunidad, contribuyen a nuestra comunidad.

En segundo lugar, sería tirar piedras contra nuestro propio tejado, porque ellos están trabajando mucho más en X de lo que nosotros lo haríamos. Así que nuestra versión sería inferior a la suya y la gente no la usaría, lo cual significa, ¿para qué tomarse tanto trabajo?

Así que cuando alguien escribe una mejora de X Window, lo que digo es que esa persona debería cooperar con el equipo de desarrollo de X. Enviársela y dejar que la usen a su manera, porque están desarrollando un elemento muy importante de software libre. Es beneficioso para nosotros que cooperemos con ellos.

PREGUNTA. Excepto, teniendo en cuenta X, en concreto... hace unos dos años el Consorcio X estaba muy metido en el movimiento *open source*. . .

STALLMAN. Bueno, en realidad no era *open source*. Quizás dijeron que lo era. No recuerdo si lo dijeron o no. Pero no era *open source*. Era restringido. No podías distribuirlo comercialmente, creo. O no podías distribuir comercialmente una versión modificada, o algo así. Existía una restricción considerada inaceptable tanto por el movimiento de software libre como por el movimiento del *open source*.

Y sí, a eso es a lo que te expones cuando usas una licencia que no es copyleft. En realidad, el Consorcio X tenía una política muy rígida. Dicen: «Si tu programa tiene copyleft, aunque sea una parte muy pequeña, no lo distribuiremos en absoluto». No lo pondremos en nuestra distribución. Así que mucha gente recibió presiones en este sentido para que no utilizara copyleft. Y el resultado fue que más tarde todo su software fue abierto, completamente. La misma gente que había presionado a los desarrolladores para que fueran muy permisivos, más tarde decían: «Vale, ahora podemos poner restricciones», lo cual no era muy ético por su parte.

Pero, dada la situación, ¿de verdad queremos reunir a duras penas los recursos necesarios para mantener una versión de X alternativa y protegida por la GPL? No tendría ningún sentido. Hay otras muchas cosas que necesitamos hacer. Hagamos esas en su lugar. Podemos cooperar con los desarrolladores de X.

PREGUNTA. ¿Podrías decirnos si GNU es una marca registrada? ¿Y si es práctico incluir como parte de la GNU GPL la autorización de marcas?

STALLMAN. En realidad, estamos solicitando un registro de marca para GNU. Pero realmente no tiene nada que ver con eso. Es una historia muy larga de explicar.

PREGUNTA. Podríais obligar a que la marca apareciera en los programas protegidos por la GPL.

STALLMAN. No, no lo creo. Las licencias protegen programas individuales. Y cuando un programa dado es parte del proyecto GNU, nadie miente al respecto. El nombre del sistema como un todo es un asunto diferente. Y esto es un tema aparte. No merece la pena discutir más sobre ello.

PREGUNTA. Si pudieras apretar un botón para obligar a todas las empresas a hacer libre su software, ¿lo harías?

STALLMAN. Bueno, sólo lo usaría con el software publicado. Creo que la gente tiene derecho de escribir un programa privado y usarlo. Y eso incluye a las empresas. Esto es un asunto de privacidad. Y es cierto, puede haber ocasiones en que esto puede ser incorrecto, como en el caso de que sea tremendamente útil para la humanidad y se lo estés ocultando. Está mal pero es un tipo de mal diferente. Es un asunto diferente, aunque se encuentre en el mismo ámbito.

Pero sí, pienso que todo el software publicado debería ser software libre. Y recordad, cuando no es software libre, se debe a la intervención gubernamental. El gobierno está interviniendo para hacerlo no libre. El gobierno está creando poderes legales especiales para hacer concesiones a los propietarios de programas, de modo que puedan tener a la policía impidiendo que usemos los programas de cierta forma. Así que ciertamente me gustaría acabar con eso.

ED SCHONBERG. La exposición de Richard ha generado una cantidad enorme de energía intelectual. Me gustaría recomendar que parte de ella se utilizara para usar, y posiblemente escribir, software libre.

Deberíamos cortar pronto el debate. Quiero decir que Richard ha inyectado en una profesión, que es conocida entre el público general por su terminal ineptitud política, un nivel de debate político y moral que, creo, no tiene precedentes en nuestra profesión. Le debemos mucho por ello. [Aplausos del público].

Capítulo 21

Algunas palabras y frases confusas que vale la pena evitar[1]

Existen ciertas palabras y frases que recomendamos evitar, al menos en ciertos contextos o con ciertos usos, bien porque son ambiguas o bien porque expresan indirectamente una opinión con la que esperamos que no estás totalmente de acuerdo.

Comercial

Por favor no uses «comercial» como sinónimo de «no libre». Se estarían confundiendo dos asuntos completamente distintos.

Un programa es comercial si se desarrolla como parte de una actividad empresarial. Un programa comercial puede ser libre o no libre, según su licencia. De la misma forma, un programa desarrollado por una escuela o un particular puede ser libre o no libre, según su licencia. Ambas cuestiones, qué clase de entidad desarrolló el programa y qué libertad tienen sus usuarios, son independientes.

En la primera década del Movimiento del Software Libre, los paquetes de software libre eran casi siempre no comerciales; los componentes del sistema operativo GNU/Linux fueron desarrollados por individuos u organizaciones sin ánimo de lucro como la FSF y algunas universidades. Pero en la década de 1990 comenzó a aparecer software libre comercial.

El software libre comercial es una contribución a nuestra comunidad, por lo que debemos promoverlo. Pero quienes piensen que «comercial» significa «no libre» tenderán a pensar que la combinación de «libre» y «comercial» es contradictoria, y rechazarán esa posibilidad. Tengamos cuidado de no utilizar la palabra «comercial» de esa manera.

Contenido

Para describir un estado de comodidad y satisfacción, sin ninguna duda dirás que estás «contento», pero es mejor no usar esta palabra para describir obras escritas y otras

[1] Escrito originalmente en 1996.

obras de autoría, ya que encierra una actitud específica hacia esas mismas obras:[2] que son un producto intercambiable cuyo fin es el de llenar un recipiente y ganar dinero. En realidad, es una falta de respeto hacia las propias obras.

Los que usan este término suelen ser a menudo los editores que presionan para que aumente el poder del copyright en nombre de los autores —«creadores», como ellos dicen— de las obras. El término «contento» revela cómo se sienten realmente.

Ya que hay quien utiliza la expresión «proveedor de contenidos», los disidentes políticos bien pueden autodenominarse «proveedores descontentos».

Creador

El término «creador», aplicado a los autores, los equipara implícitamente a una deidad («el Creador»). Este término es utilizado por los editores para elevar la estatura moral de los autores por encima de la del ciudadano medio, justificando un mayor poder del copyright que los editores pueden ejercer en nombre de los autores.

Freeware

Por favor no uses el término «freeware» como sinónimo de «software libre». El término «freeware» fue utilizado con frecuencia durante la década de 1980 para hacer referencia a programas publicados sólo como ejecutables, con el código fuente no disponible. Hoy en día no está aceptada ninguna definición específica.

De la misma manera, si utilizas otro idioma diferente al inglés, por favor intenta evitar palabras tomadas del inglés como «free software» o «freeware». Intenta utilizar palabras menos ambiguas que te ofrezca tu propio idioma, por ejemplo:

- Alemán: *freie software*
- Árabe: *baramej horrah*
- Catalán: *Programari lliure*
- Checo: *svobodný software*
- Chino: *ziyou ruan jian*
- Danés: *fri software* (o también: *frit programmel*)
- Eslovaco: *slobodny softver*
- Esloveno: *prosto programje*
- Español: *software libre*

[2] En inglés la palabra *content* significa tanto «contenido» como «contento» o «satisfecho», circunstancia que Stallman emplea para hacer un juego de palabras en esta entrada. [*N. del E.*]

- Esperanto: *libera softvaro*
- Finés: *vapaa ohjelmiston*
- Francés: *logiciel libre*
- Hebreo: *tochna chofshit*
- Húngaro: *szabad szoftver*
- Italiano: *software libero*
- Irlandés: *bog earraí saoired*
- Japonés: *jiyuu-na software*
- Neerlandés: *vrije software*
- Portugués: *software livre*
- Rumano: *software liber*
- Sueco: *fri programvara*
- Tamil: *Sudanthiram software*
- Turco: *õzgür yazilim*

Gestión de derechos digitales (DRM)

El software para la gestión de derechos digitales (*Digital Rights Management* o DRM) está diseñado en realidad para imponer restricciones a los usuarios de ordenadores. El uso de la palabra «derechos» en esta expresión responde a una propaganda pensada para que, sin darte cuenta, veas el asunto desde el punto de vista de los pocos que imponen las restricciones, mientras ignoras los derechos de los muchos a quienes las restricciones son impuestas.

Son buenas alternativas: «Gestión de restricciones digitales» o «software con grilletes».

Licencia de tipo BSD

La expresión «licencia de tipo BSD» lleva a confusión porque *mezcla* licencias que tienen importantes diferencias. Por ejemplo, la licencia BSD original con la cláusula sobre publicidad es incompatible con la GNU GPL, pero la licencia BSD revisada sí es compatible con la GPL.

Para evitar confusiones es mejor nombrar *la licencia específica en cuestión* y evitar la vaga expresión «de tipo BSD».

Piratería

Los editores frecuentemente se refieren a la copia prohibida como "piratería". De esta forma, expresan indirectamente que hacer copias ilegales es éticamente equivalente a atacar barcos en alta mar, secuestrar y asesinar a la gente que viaja en ellos.

Si no crees que la copia ilegal sea como secuestrar y asesinar, podrías preferir no usar la palabra *piratería* para describirla. Expresiones neutrales como «copia prohibida» o «copia no autorizada» pueden utilizarse en su lugar. Incluso algunos de nosotros podríamos preferir utilizar una expresión positiva tal como çompartir información con tu vecino".

Propiedad intelectual

A los editores y a los abogados les gusta describir el «copyright» como «propiedad intelectual». La expresión «propiedad intelectual» conlleva una presunción oculta: que la forma más normal de pensar respecto a la cuestión de la copia se basa en una analogía con los objetos físicos y en la idea que nos hacemos de ellos como propiedad.

Pero esta analogía pasa por alto la diferencia crucial entre objetos materiales e información: la información puede ser copiada y compartida casi sin esfuerzo, mientras que los objetos materiales no. Fundar tu pensamiento en esta analogía es ignorar esta diferencia.

Incluso el sistema legal de los EEUU no acepta por completo esta analogía, debido a que no trata al copyright como derechos de propiedad sobre objetos físicos.

Si no quieres limitarte en esta forma de pensar, es mejor evitar la expresión «propiedad intelectual» en tus palabras y en tus pensamientos.

Hay otro problema con el término «propiedad intelectual»: es un término genérico en el que se mezclan varios sistemas legales diferentes, incluyendo copyrights, patentes, marcas registradas y otros, que tienen muy poco en común. Estos sistemas legales, que se originaron por separado, regulan actividades distintas, operan de distinta manera y plantean distintas cuestiones sobre las normativas públicas. Por ejemplo, si aprendes algo sobre las leyes de copyright, harás bien en asumir que no es una ley de patentes, ya que casi siempre es así. Puesto que estas leyes son tan distintas, la expresión «propiedad intelectual» es una invitación a una generalización simplista. Así, cualquier opinión sobre «propiedad intelectual» será, casi con seguridad, una estupidez. En un contexto tan impreciso, ni siquiera se pueden apreciar los problemas específicos sobre política pública que plantean las leyes de copyright o los diferentes problemas suscitados por las leyes de patentes, o por cualquiera de las otras.

El término «propiedad intelectual» lleva a la gente a centrarse en el mínimo común denominador de estas leyes distintas, que es el hecho de que establecen algunas abstracciones que pueden comprarse y venderse, e ignorar el aspecto importante, que son las restricciones que imponen al público y qué bien o mal provocan estas mismas restricciones.

Si quieres pensar con claridad en las cuestiones que plantean las patentes, el copyright y las marcas registradas, o incluso aprender qué dictan estas leyes, el primer paso consiste en olvidarte de que alguna vez oíste la expresión «propiedad intelectual» y tratar cada una de las cuestiones de forma independiente. Para ofrecer una información clara y fomentar un pensamiento lúcido, nunca hables o escribas sobre «propiedad intelectual»; en lugar de eso, expón el tema de manera concreta sobre el copyright, las patentes o cualquier otra ley específica a la que te estás refiriendo.

De acuerdo con el profesor Mark Lemley de la Facultad de Derecho de la Universidad de Tejas, el extendido uso de la expresión «propiedad intelectual» es una manía reciente, surgida en 1967 en la fundación de la Organización Mundial de la Propiedad Intelectual[3] (OMPI, o WIPO, en sus siglas en inglés). La OMPI defiende los intereses de los titulares del copyright, patentes y marcas registradas, presionando a los gobiernos para aumentar su poder. Uno de los tratados de la OMPI sigue la línea de la Digital Millenium Copyright Act que se ha utilizado en los Estados Unidos para censurar algunos paquetes muy útiles de software libre en EE.UU.[4]

Protección

A los abogados de los editores les encanta utilizar el término «protección» para describir el «copyright». Esta palabra lleva implícita la idea de que evita la destrucción o el sufrimiento; por lo tanto, impulsa a la gente a identificarse con el propietario y el editor, que se benefician del «copyright», en lugar de identificarse con los usuarios que son restringidos por él.

Es fácil evitar el término «protección» y utilizar expresiones neutrales en su lugar. Por ejemplo, en lugar de «la protección del copyright permanece vigente por un tiempo prolongado», puedes decir, «el copyright permanece vigente por un tiempo prolongado».

Si quieres criticar el copyright en lugar de apoyarlo, puedes emplear la expresión «restricciones del copyright».

RAND (razonable y no discriminatoria)

Los organismos de estandarización, que promulgan estándares restringidos de patentes que prohíben el software libre, suelen llevar a cabo una política para la obtención de licencias de patentes que requiere el pago de una tasa fija por cada copia hecha de un programa que cumpla esos estándares. A menudo se refieren a dichas licencias con el término «RAND», que son las siglas de *Reasonable And Non-Discriminatory* («razonable y no discriminatoria»).

[3]Véase la nota 123 de su reseña de marzo de 1997, en la revista de Derecho de Tejas, de *Romantic Autorship and the Rhetoric of Property* de James Boyle.

[4]Visita `http://www.wipout.net` para informarte sobre una campaña contra la OMPI.

Este término no es más que un lavado de cara para un tipo de licencias de patentes que no son normalmente ni razonables ni no discriminatorias. Es cierto que estas licencias no discriminan a ninguna persona en particular, pero sí discriminan a la comunidad del software libre, lo que no las hace en absoluto razonables. Así que la mitad de RAND es engañosa y la otra mitad discriminatoria.

Los organismos de estandarización deberían reconocer que estas licencias son discriminatorias y no usar la expresión «razonable y no discriminatoria» o RAND para describirlas. Hasta que lo reconozcan, los escritores que no quieran unirse a ese lavado de cara harán bien en descartar esta expresión. Aceptarla y utilizarla meramente porque las compañías esgrime-patentes han extendido su uso, es permitir que esas compañías dicten su propia voluntad.

Recomiendo la expresión *uniform fee only*, o UFO de forma más abreviada,[5] como alternativa. Esta es la expresión exacta, ya que la única condición de estas licencias es el pago de una tasa uniforme de uso.

Robo

Los apologetas del copyright emplean con frecuencia palabras como *robo* y *hurto* para describir la violación del copyright. Al mismo tiempo, piden que consideremos el sistema legal como una autoridad ética: si copiar está prohibido, debe ser malo. De este modo, es pertinente mencionar que el sistema legal —al menos en los Estados Unidos— rechaza la idea de que la violación del copyright sea un «robo». Los apologetas del copyright apelan a la autoridad, mientras tergiversan lo que la propia autoridad dice. La idea de que las leyes deciden qué está bien o qué esta mal responde normalmente a una equivocación. Las leyes son, en el mejor de los casos, un intento de alcanzar justicia; decir que las leyes definen la justicia o la conducta ética es darle la vuelta a las cosas.

Software gratuito

Si quieres decir que un programa es software libre, por favor no digas que está disponible «de forma gratuita». Esa expresión significa explícitamente que tiene un «precio cero». El software libre es una cuestión de libertad, no de precio.

Las copias de software libre frecuentemente están disponibles de forma gratuita —por ejemplo, para ser descargadas por FTP. Pero las copias de software libre también están disponibles por un cierto precio en CD-ROM; también las copias de software propietario están disponibles ocasionalmente en promoción de forma gratuita y algunos paquetes propietarios están en muchas ocasiones disponibles, sin cobrar nada, para ciertos usuarios.

Para evitar la confusión, puedes decir que el programa está disponible «como software libre».

[5] *Uniform fee only:* «sólo tasa uniforme». UFO son las siglas en inglés de *unidentified flying object*, «objeto volante no identificado». [*N. del E.*]

Software regalado

Es erróneo utilizar el término «regalar» para referirse a «distribuir un programa como software libre». Tiene el mismo problema que «software gratuito»: expresa que lo importante es el precio, no la libertad. Una forma de evitar esta confusión es decir «publicado como software libre».

Vender software

La expresión «vender software» es ambigua. Estrictamente hablando, intercambiar una copia de un programa libre por una cantidad de dinero es «vender»; pero la gente normalmente asocia el término «vender» con restricciones propietarias en el uso consecutivo del software. Puedes ser más preciso y evitar confusiones, diciendo «distribuir copias de un programa por una cantidad» o «imponer restricciones propietarias al uso de un programa», dependiendo de lo que quieras expresar.

Véase el capítulo «vender software libre» para profundizar sobre esta cuestión.

Parte IV

Licencias

Apéndice A
Licencia Pública General GNU[1]

General Public License o GPL-GNU

Versión 2, junio 1991,

Preámbulo

Las licencias que cubren la mayor parte del software están diseñadas para despojarle de la libertad para compartirlo y para modificarlo. Por el contrario, la Licencia Pública General de GNU pretende garantizar la libertad de compartir y modificar software libre —para asegurar que el software es libre para todos sus usuarios. Esta Licencia Pública General se aplica a la mayor parte del software de la *Free Software Foundation* y a cualquier otro programa si sus autores se comprometen a utilizarla. [Existe otro software de la *Free Software Foundation* que está cubierto por la Licencia Pública General de GNU para Bibliotecas.] También puedes aplicarla a sus propios programas.

Cuando hablamos de software libre, estamos refiriéndonos a la libertad, no al precio. Nuestra Licencia Pública General está diseñada para asegurarnos de que tenga la libertad de distribuir copias de software libre —y cobrar por ese servicio si quiere—, de que reciba el código fuente o de que pueda conseguirlo si así lo desea, de que pueda modificar el software o utilizar fragmentos del mismo en nuevos programas libres, y de que sepa que puede hacer todas estas cosas.

[1]This is an unofficial translation of the GNU General Public License into spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU LGPL —only the original English text of the GNU GPL does that. However, we hope that this translation will help spanish speakers understand the GNU GPL better. [Esta es una traducción no oficial al español de la GNU General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU GPL. Estas condiciones se establecen solamente por el texto original, en inglés, de la GNU GPL. Sin embargo, esperamos que esta traducción ayude a los hispanohablantes a entender mejor la GNU GPL.]

Para proteger sus derechos, necesitamos algunas restricciones que prohíban negarle a usted estos derechos o pedirle que renuncie a ellos. Estas restricciones se traducen en ciertas obligaciones que le afectan si distribuye copias del software, o si modifica software.

Por ejemplo, si distribuye copias de uno de estos programas, ya sea gratuitamente, o a cambio de unos honorarios, debe dar a los receptores todos los derechos que posee. Debe asegurarse de que ellos también reciben, o pueden conseguir, el código fuente. Y debe mostrarles estas condiciones de forma que conozcan sus derechos.

Protegemos sus derechos por medio de la combinación de dos medidas: (1) ponemos el software bajo copyright y (2) le ofrecemos esta licencia, que le da permiso legal para copiar, distribuir y/o modificar el software.

También, para proteger a cada autor y a nosotros mismos, queremos asegurarnos de que todo el mundo comprende que no se proporciona ninguna garantía para este software libre. Si el software es modificado y distribuido, queremos que sus receptores sepan que lo que tienen no es el original, de forma que cualquier problema introducido por otros no afecte a la reputación de los autores originales.

Por último, cualquier programa libre está constantemente amenazado por las patentes de software. Queremos evitar el peligro de que los distribuidores de un programa libre lo patenten por su cuenta, convirtiendo así el programa en propietario. Para evitar esto, hemos dejado claro que cualquier patente debe ser registrada para el libre uso, o no ser registrada de ninguna manera.

Los términos exactos y las condiciones para la copia, distribución y modificación se exponen a continuación.

Términos y condiciones para la copia, distribución y modificación de la Licencia Pública General de GNU

0. Esta licencia se aplica a cualquier programa u otro tipo de trabajo que contenga una nota colocada por el titular del copyright que señale que puede ser distribuido bajo los términos de esta Licencia Pública General. En adelante, el «Programa» se referirá a cualquier programa o trabajo que cumpla esa condición y el «trabajo basado en el programa» se referirá bien al Programa o a cualquier trabajo derivado de él según la ley de copyright; esto es, un trabajo que contenga el programa o una porción de él, ya sea de forma literal o con modificaciones y/o traducido en otro lenguaje. (Por lo tanto, la traducción está incluida sin limitaciones en el término «modificación».) Cada titular de la licencia será denominado «usted».

   Cualquier otra actividad que no sea la copia, distribución o modificación no está cubierta por esta licencia y está fuera de su incumbencia. El acto de ejecutar el Programa no está restringido y los resultados del Programa están cubiertos únicamente si sus contenidos constituyen un trabajo basado en el Programa, independientemente de haberlo producido mediante la ejecución del programa. El que esto se cumpla, depende de lo que haga el Programa.

1. Usted puede copiar y distribuir copias literales del código fuente del Programa, según lo ha recibido, en cualquier medio, siempre que de forma adecuada y bien visible publique en cada copia un anuncio de copyright adecuado y un repudio de garantía; mantenga intactos todos los anuncios que se refieran a esta licencia y a la ausencia de garantía y proporcione a cualquier otro receptor del programa una copia de esta licencia junto con el Programa.

   Puede cobrar un precio por el acto físico de transferir una copia, y puede, según su libre albedrío, ofrecer una garantía a cambio de unos honorarios.

2. Puede modificar su copia o sus copias del Programa o de cualquier fragmento del mismo, creando de esta manera un trabajo basado en el Programa, y puede copiar y distribuir esa modificación o trabajo bajo los términos del apartado 2, antedicho, siempre que además cumpla las siguientes condiciones:

   a. Debe hacer que los ficheros modificados lleven anuncios prominentes indicando que los ha cambiado y la fecha de cualquier modificación.
   b. Debe hacer que cualquier trabajo que distribuya o publique y que en todo o en parte contenga o sea derivado del Programa o de cualquier parte de él sea registrado como un todo, sin carga alguna a terceras partes bajo las condiciones de esta licencia.
   c. Si el programa modificado lee normalmente órdenes interactivamente cuando es ejecutado, debe hacer que, cuando comience su ejecución para ese uso interactivo de la forma habitual, muestre o escriba un mensaje que incluya un anuncio del copyright y un anuncio de que no se ofrece ninguna garantía —o por el contrario que usted provee la garantía— y que los usuarios pueden redistribuir el programa bajo estas condiciones, indicando al usuario cómo puede ver una copia de esta licencia. (Excepción: si el propio programa es interactivo pero normalmente no muestra este anuncio, no se requiere que su trabajo basado en el Programa muestre ningún anuncio.)

   Estos requisitos se aplican al trabajo modificado como un todo. Si partes identificables de ese trabajo no son derivadas del Programa, y pueden, razonablemente, considerarse trabajos independientes y separados en sí mismos, entonces esta Licencia y sus términos no se aplicarán a esas partes cuando sean distribuidas como trabajos separados. Pero cuando distribuya esas mismas secciones como partes de un conjunto, que no deja de ser un trabajo basado en el Programa, la distribución del conjunto debe hacerse según los términos de esta licencia, cuyos permisos para otros propietarios de la licencia se extienden al conjunto completo, y por lo tanto a todas y cada una de sus partes, con independencia de quién la escribió.

   Por lo tanto, no es la intención de este apartado reclamar derechos o desafiar sus derechos sobre trabajos escritos completamente por usted. Más bien se intenta ejercer el derecho a controlar la distribución de los trabajos derivados o colectivos basados en el Programa.

Además, el simple hecho de reunir un trabajo no basado en el Programa con el Programa —o con un trabajo basado en el Programa— en un volumen de almacenamiento o en un medio de distribución no hace que dicho trabajo entre dentro del ámbito cubierto por esta licencia.

3. Puede copiar y distribuir el Programa —o un trabajo basado en él, según se especifica en el apartado 3—, como código objeto o en formato ejecutable según los términos de los apartados 2 y 3, siempre que además cumpla una de las siguientes condiciones:

    a. Acompañarlo con el código fuente completo correspondiente, en formato electrónico, que debe ser distribuido según se especifica en los apartados 2 y 3 de esta Licencia en un medio habitualmente utilizado para el intercambio de programas, o
    b. Acompañarlo con una oferta por escrito, válida al menos durante tres años, para proporcionar a terceros una copia completa en formato electrónico del código fuente correspondiente, a un coste no mayor que el de realizar físicamente la distribución del código fuente, que será distribuido bajo las condiciones descritas en los apartados 2 y 3, en un medio habitualmente utilizado para el intercambio de programas, o
    c. Acompañarlo con la información que recibió ofreciendo distribuir el código fuente correspondiente. (Esta opción se permite sólo para distribución no comercial y sólo si usted recibió el programa como código objeto o en formato ejecutable con tal oferta, de acuerdo con el apartado b anterior.)

    Por código fuente de un trabajo se entiende la forma preferida del trabajo cuando se introducen modificaciones. Para un trabajo ejecutable, se entiende por código fuente completo el código fuente de todos los módulos que contiene, además de cualquier fichero asociado de definición de interfaces y de los guiones utilizados para controlar la compilación e instalación del ejecutable. Como excepción especial, el código fuente distribuido no necesita incluir nada que sea distribuido normalmente —bien como fuente, bien en forma binaria— con los componentes principales —compilador, kernel y similares— del sistema operativo en el cual funciona el ejecutable, a no ser que el propio componente acompañe al ejecutable.

    Si la distribución del ejecutable o del código objeto se hace mediante la oferta de un acceso para copiarlo de un cierto lugar, entonces se considera la oferta de acceso para copiar el código fuente de ese mismo lugar como distribución del código fuente, incluso aunque terceras partes no estén forzadas a copiar la fuente junto con el código objeto.

4. No puede copiar, modificar, «sublicenciar» o distribuir el Programa excepto como prevé expresamente esta licencia. Cualquier intento de copiar, modificar, «sublicenciar» o distribuir el Programa de otra forma no es válida, y hará que cesen

automáticamente los derechos que le proporciona esta Licencia. En cualquier caso, las partes que hayan recibido copias o derechos de usted bajo esta Licencia no cesarán en sus derechos mientras esas partes continúen cumpliéndola.

5. No está obligado a aceptar esta licencia, ya que no la ha firmado. Sin embargo, no hay nada más que le autoriza a modificar o distribuir el Programa o sus trabajos derivados. Estas acciones están prohibidas por la ley si no acepta esta Licencia. Por lo tanto, si modifica o distribuye el Programa —o cualquier trabajo basado en el Programa—, está indicando que acepta esta Licencia para poder hacerlo, y todos sus términos y condiciones para copiar, distribuir o modificar el Programa o trabajos basados en él.

6. Cada vez que redistribuya el Programa —o cualquier trabajo basado en él—, el receptor recibe automáticamente una licencia del emisor de la licencia original para copiar, distribuir o modificar el Programa, sujeta a estos términos y condiciones. No puede imponer al receptor ninguna restricción adicional sobre el ejercicio de los derechos garantizados aquí. Usted no es responsable de hacer cumplir esta licencia a terceros.

7. Si, como consecuencia de una resolución judicial o de una alegación de infracción de patente o por cualquier otra razón —no limitada a asuntos relacionados con patentes—, se le impusieran condiciones —ya sea por mandato judicial, por acuerdo o por cualquier otra causa— que contradigan las condiciones de esta licencia, esto no le exime de cumplir las condiciones de la misma. Si no puede distribuir el Programa de forma que se satisfagan simultáneamente sus obligaciones bajo esta licencia y cualquier otra obligación pertinente, entonces no podrá distribuir el Programa de ninguna forma. Por ejemplo, si una patente no permite la redistribución libre de derechos de autor del Programa por parte de todos aquellos que reciban copias directa o indirectamente a través de usted, entonces la única forma en que podría satisfacer tanto esa condición como esta licencia sería evitar completamente la distribución del Programa.

   Si cualquier porción de este apartado se considera nula o imposible de cumplir en cualquier circunstancia particular habrá de cumplirse el resto y la sección por entero habrá de cumplirse en cualquier otra circunstancia.

   No es el propósito de este apartado inducirle a infringir ninguna reivindicación de patente ni de ningún otro derecho de propiedad o a impugnar la validez de ninguna de dichas reivindicaciones. Este apartado tiene el único propósito de proteger la integridad del sistema de distribución de software libre, que se realiza mediante prácticas de licencia pública. Mucha gente ha hecho contribuciones generosas a la amplia variedad de software distribuido mediante ese sistema, con la confianza de que el sistema se aplicará consistentemente. Pertenece al autor/donante decidir si quiere distribuir software mediante cualquier otro sistema; una licencia no puede imponer esa elección.

Este apartado pretende dejar completamente claro lo que se cree que es una consecuencia del resto de esta Licencia.

8. Si la distribución y/o uso del Programa está restringida en ciertos países, ya sea por medio patentes o por interfaces bajo copyright, el titular del copyright que coloca este Programa bajo esta licencia puede añadir una limitación explícita de distribución geográfica excluyendo dichos países, de forma que la distribución se permita sólo en o entre los países no excluidos de esta manera. En ese caso, esta licencia incorporará la limitación como si estuviese escrita en el cuerpo de esta licencia.

9. La *Free Software Foundation* puede publicar versiones revisadas y/o nuevas de la Licencia Pública General de cuando en cuando. Estas nuevas versiones serán similares en espíritu a la presente versión, pero pueden ser diferentes en algunos detalles con el fin considerar nuevos problemas o situaciones.

   Cada versión recibe un número que la distingue de otras. Si el Programa especifica un número de versión de esta licencia que se refiere a ella y a «cualquier versión posterior», tiene la opción de seguir los términos y condiciones, bien de esa versión, bien de cualquier versión posterior publicada por la *Free Software Foundation*. Si el Programa no especifica un número de versión para esta licencia, podrás escoger cualquier versión publicada por la *Free Software Foundation*.

10. Si quisiera incorporar ciertas partes del Programa en otros programas libres cuyas condiciones de distribución son diferentes, contacte al autor para pedirle permiso. Si el software tiene copyright de la *Free Software Foundation*, escriba a la *Free Software Foundation*: algunas veces hacemos excepciones en estos casos. Nuestra decisión estará guiada por el doble objetivo de preservar la libertad de todos los derivados de nuestro software libre y de promover que se comparta y reutilice el software en general.

AUSENCIA DE GARANTÍA

11. DADO QUE EL PROGRAMA SE LICENCIA DE FORMA GRATUITA, NO SE OFRECE NINGUNA GARANTÍA SOBRE EL PROGRAMA EN TODA LA EXTENSIÓN PERMITIDA POR LA LEGISLACIÓN APLICABLE. EXCEPTO CUANDO SE INDIQUE DE OTRA FORMA POR ESCRITO, LOS TITULARES DEL COPYRIGHT Y/U OTRAS PARTES PROPORCIONAN EL PROGRAMA «TAL CUAL», SIN GARANTÍA DE NINGUNA CLASE, YA SEA EXPRESA O IMPLÍCITA, INCLUYENDO —PERO NO LIMITADO POR— LAS GARANTÍAS MERCANTILES IMPLÍCITAS O A LA CONVENIENCIA PARA CUALQUIER PROPÓSITO PARTICULAR. CUALQUIER RIESGO REFERENTE A LA CALIDAD Y A LAS PRESTACIONES DEL PROGRAMA ES ASUMIDO POR USTED. SI SE PROBASE QUE EL PROGRAMA ES DEFECTUOSO, ASUME EL COSTE DE CUALQUIER SERVICIO, REPARACIÓN O CORRECCIÓN.

12. EN NINGÚN CASO, SALVO QUE LO REQUIERA LA LEGISLACIÓN APLICABLE O HAYA SIDO ACORDADO POR ESCRITO, NINGÚN TITULAR DEL COPYRIGHT NI NINGUNA OTRA PARTE QUE MODIFIQUE Y/O REDISTRIBUYA EL PROGRAMA SEGÚN SE PERMITE EN ESTA LICENCIA SERÁ RESPONSABLE ANTE USTED POR DAÑOS, INCLUYENDO CUALQUIER DAÑO GENERAL, ESPECIAL, INCIDENTAL O RESULTANTE PRODUCIDO POR EL USO O LA IMPOSIBILIDAD DE USO DEL PROGRAMA — INCLUYENDO, PERO NO LIMITADO POR, LA PÉRDIDA DE DATOS, LA GENERACIÓN INCORRECTA DE DATOS, LAS PÉRDIDAS SUFRIDAS POR USTED O POR TERCEROS, Y UN FALLO DEL PROGRAMA AL FUNCIONAR EN COMBINACIÓN CON CUALQUIER OTRO PROGRAMA—, INCLUSO SI DICHO TITULAR U OTRA PARTE HA SIDO ADVERTIDO DE LA POSIBILIDAD DE DICHOS DAÑOS.

FIN DE TÉRMINOS Y CONDICIONES

Apéndice. Cómo aplicar estos términos a sus nuevos programas.

Si usted desarrolla un nuevo Programa, y quiere que sea del mayor uso posible para el público en general, la mejor forma de conseguirlo es convirtiéndolo en software libre que cualquiera pueda redistribuir y cambiar según estos términos.

Para hacerlo, añada las siguientes cláusulas al programa. Lo más seguro es añadirlas al principio de cada fichero fuente para transmitir lo más efectivamente posible la ausencia de garantía. Además, cada fichero debería tener al menos la línea de «copyright» y un indicador de dónde puede encontrarse el anuncio completo.

<una línea para indicar el nombre del programa y una breve idea de qué hace.>

```
Copyright (C) 19aa <nombre del autor>

Este programa es software libre. Puede redistribuirlo y/o
modificarlo bajo los términos de la Licencia Pública General
de GNU según es publicada por la Free Software Foundation,
bien de la versión 2 de dicha Licencia o bien --según su
elección-- de cualquier versión posterior.

Este programa se distribuye con la esperanza de que sea
útil, pero SIN NINGUNA GARANTÍA, incluso sin la garantía
MERCANTIL implícita o sin garantizar la CONVENIENCIA PARA UN
PROPÓSITO PARTICULAR. Para más detalles, véase la Licencia
Pública General de GNU.

Debería haber recibido una copia de la Licencia Pública
General junto con este programa. En caso contrario, escriba
a la Free Software Foundation, Inc., en 675 Mass Ave,
Cambridge, MA 02139, EEUU.
```

Añada también información sobre cómo contactar con usted mediante correo electrónico y postal.

Si el programa es interactivo, haga que muestre un pequeño anuncio como el siguiente, cuando comienza a funcionar en modo interactivo:

```
Gnomovision versión 69, Copyright © 19aa nombre del autor

Gnomovision no ofrece ABSOLUTAMENTE NINGUNA GARANTÍA. Para
más detalles escriba «show c».
```

Los comandos hipotéticos «show w» y «show c» deberían mostrar las partes adecuadas de la Licencia Pública General. Por supuesto, los comandos que use pueden llamarse de cualquier otra manera. Podrían incluso ser pulsaciones del ratón o elementos de un menú —lo que sea apropiado para su programa.

También deberá conseguir que su empleador —si trabaja como programador— o su Universidad —si es el caso— firme un «renuncia de copyright» para el programa, en caso de que sea necesario. A continuación se ofrece un ejemplo, altere los nombres según sea conveniente:

```
Yoyodyne, Inc. mediante este documento renuncia a cualquier
interés de derechos de copyright con respecto al programa
Gnomovision --que hace pasadas a compiladores-- escrito por
James Hacker

<firma de Ty Coon>, 20 de diciembre de 1996

Ty Coon, Presidente de Asuntillos Varios.
```

Esta Licencia Pública General no permite incorporar sus programas en programas propietarios. Si su programa es una biblioteca de subrutinas, puede considerar más útil permitir enlazar aplicaciones propietarias con la biblioteca. Si este es el caso, use la Licencia Pública General de GNU para Bibliotecas en lugar de esta Licencia.

Apéndice B

Licencia Pública General Menor[1]

GNU Lesser General Public Licence, GNU-GPL

Versión 2.1, Febrero 1999

Preámbulo

Las licencias, para la mayoría del software, están diseñadas para eliminar la libertad de compartirlo o de modificarlo. En contraste, las Licencias Públicas Generales GNU pretenden garantizar la libertad de compartir y modificar el software; para asegurar que el software sea libre para todos los usuarios.

Esta licencia, la Licencia Pública General Menor [Lesser General Public Licence], se aplica a ciertos paquetes de software específicamente diseñados —normalmente bibliotecas— de la *Free Software Foundation* y de otros autores que deciden usarla. Usted también puede usarla, pero le sugerimos que piense primero cuidadosamente si esta licencia o la General Public Licence ordinaria, es o no la mejor estrategia en su caso particular, basándose en las explicaciones siguientes.

Cuando hablamos de software libre, nos referimos a libertad de uso, no al precio. Nuestra Licencia Pública General está diseñada para asegurar que usted sea libre de distribuir copias de software libre —y cobrar por este servicio si así lo desea—; que

[1]This is an unofficial translation of the GNU Lesser General Public License into spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms for software that uses the GNU LGPL —only the original English text of the GNU LGPL does that. However, we hope that this translation will help spanish speakers understand the GNU GPL better. [Esta es una traducción no oficial al español de la GNU Lesser General Public License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución para el software que usa la GNU LGPL. Estas condiciones se establecen solamente por el texto original, en inglés, de la GNU LGPL. Sin embargo, esperamos que esta traducción ayude a los hispanohablantes a entender mejor la GNU LGPL.] Esta es la primera versión emitida de la GPL Menor. Es también la sucesora de la GNU Library Public Licence, versión 2, de aquí la versión 2.1

reciba el código fuente o que pueda obtenerlo si así lo quiere; que pueda modificar el software y utilizar partes de él en nuevos programas libres; y que haya sido informado de que tiene estos derechos.

Para proteger sus derechos, necesitamos imponer restricciones que prohíban a los distribuidores negarle estos derechos o a pedirle que renuncie a los mismos. Estas restricciones se traducen en ciertas responsabilidades si distribuye copias de la biblioteca o si la modifica.

Por ejemplo, si distribuye copias de la biblioteca, de forma gratuita o no, debe ofrecer a los receptores todos los derechos que nosotros le ofrecemos a usted. Debe asegurarse de que ellos también reciban o puedan obtener el código fuente. Si usted enlaza otro código con la biblioteca, debe proporcionar a los receptores los ficheros objeto completos, de forma que ellos puedan reenlazarlos con la biblioteca después de introducir cambios en la biblioteca y recompilarla. Y debe mostrarles estos términos para que conozcan sus derechos.

Nosotros protegemos sus derechos con un método que consiste en dos pasos: (1) obtenemos los derechos de autor de la biblioteca y (2) le ofrecemos esta licencia que le autoriza acopiar, distribuir y/o modificar la biblioteca.

Para proteger a los distribuidores, queremos dejar muy claro que no existe garantía para la biblioteca libre. Además, si la biblioteca es modificada por alguien y se transmite, los receptores deberían saber que lo que tienen no es la versión original, de forma que la reputación del autor original no se vea afectada por problemas que podrían ser introducidos por otros.

Por último, las patentes de software plantean una amenaza constante para la existencia de cualquier programa libre. Queremos asegurarnos que una empresa no pueda limitar eficazmente a los usuarios de un programa libre mediante la obtención de una licencia restrictiva de un titular de patente. Por lo tanto, insistimos en que cualquier licencia de patente obtenida para una versión de la biblioteca sea congruente con todas las libertades de uso especificadas en esta licencia.

La mayoría del software GNU, incluyendo algunas bibliotecas, está cubierto por la Licencia Pública General GNU ordinaria. Esta licencia, la Licencia Pública General Menor GNU, se aplica a ciertas bibliotecas y es bastante diferente de la Licencia Pública General ordinaria. Usamos esta licencia para ciertas bibliotecas con el objeto de permitir el enlace de las mismas dentro de programas no libres.

Cuando un programa se enlaza con una biblioteca, ya sea estáticamente ya sea usando una biblioteca compartida, la combinación de los dos es, legalmente hablando, un trabajo combinado, un derivado de la biblioteca original. La Licencia Pública General ordinaria permitiría tal enlace sólo si la combinación completa ajusta sus criterios de libertad. La Licencia Pública General Menor introduce un criterio más laxo para el enlace de otro código con la biblioteca.

Denominamos a esta licencia como Licencia Pública General «Menor» porque hace «menos» que la Licencia Pública General ordinaria para proteger las libertades del usuario. También proporciona a los desarrolladores de programas libres «menos» ventajas sobre los programas no libres competidores. Estas desventajas son la razón por

la cual nosotros usamos la Licencia Pública General ordinaria para la mayoría de las bibliotecas. Sin embargo, la licencia Menor proporciona ventajas en circunstancias especiales.

Por ejemplo, en raras ocasiones puede haber una especial necesidad de fomentar al máximo posible el uso de una biblioteca determinada, de forma que esta se convierta en un estándar. Para ello, se debe permitir a los programas no libres el uso de estas bibliotecas. Un caso más frecuente es el de una biblioteca libre que desempeñe la misma función que la que realizan las bibliotecas no libres más utilizadas. En este caso, hay poco que ganar limitando la biblioteca únicamente al software libre, de manera que usamos la Licencia Pública General Menor.

En otros casos, el permiso para usar una biblioteca determinada en programas no libres aumenta el número de usuarios de gran cantidad de software libre. Por ejemplo, el permiso para utilizar la biblioteca GNU C en programas no libres posibilita a mucha más gente utilizar el sistema operativo GNU, así como su variante, el sistema operativo GNU/LINUX.

Aunque la Licencia Pública General Menor no protege tanto las libertades del usuario, asegura que el usuario de un programa que está enlazado con la biblioteca tiene la libertad y los medios para ejecutar ese programa usando una versión modificada de la biblioteca.

Los términos y las condiciones exactas para la copia, distribución y modificación se indican a continuación. Preste especial atención a la diferencia entre un «trabajo basado en la biblioteca» y un «trabajo que utiliza la biblioteca». El primero contiene código derivado de la biblioteca, mientras que el último debe estar unido a la biblioteca para ser ejecutado.

Términos y condiciones para la copia, distribución y modificación

0. El acuerdo de esta licencia se aplica a cualquier biblioteca de software u otro programa que contenga un aviso colocado por el titular de los derechos de autor u otras partes interesadas explicando que puede ser distribuido bajo los términos de esta Licencia Pública General Menor —llamada también «esta Licencia». A cada titular de permiso se le designa por «usted».

   Una «biblioteca» contiene una colección de funciones y/o datos de software, preparados para ser enlazados de una forma cómoda con programas de aplicación —que usan algunas de estas funciones y datos— para formar ejecutables.

   Por «Biblioteca» nos referiremos, en adelante, a cualquier trabajo o biblioteca de software que haya sido distribuido bajo estos términos. Un «trabajo basado en la Biblioteca» significa que, o la Biblioteca o cualquier trabajo derivado, están bajo la ley de copyright: es decir, un trabajo que contiene a la Biblioteca o a una parte de ella, ya sea de forma literal o con modificaciones y/o traducida de forma clara a

otro idioma —mas adelante se incluye la traducción sin restricción en el término «modificación».

El «código fuente» para un trabajo se refiere a la forma preferida del trabajo para hacer modificaciones en él. Para una Biblioteca, el código fuente completo se refiere a todos los códigos fuente para todos los módulos que contenga la biblioteca, además de cualquier fichero de definición de interfaz asociado y los guiones [*scripts*] asociados para controlar la compilación e instalación de la biblioteca.

Otras actividades que no sean la copia, distribución y modificación no se encuentran cubiertas por esta Licencia; se encuentran fuera de su ámbito. La opción de ejecutar un programa utilizando la Biblioteca no esta restringido y el resultado de dicho programa esta cubierto únicamente si su contenido constituye un trabajo basado en la Biblioteca —independientemente del uso de la Biblioteca como herramienta para escribirlo. Que esto sea cierto dependerá de lo que haga la Biblioteca y de lo que haga el programa que utiliza la Biblioteca.

1. Puede copiar y distribuir copias literales del código fuente completo de la Biblioteca tal y como lo recibe, en cualquier medio, a condición de que usted publique de forma manifiesta y apropiada, en cada una de las copias, un aviso conveniente de copyright y una renuncia de garantía; mantenga intactas todas las notificaciones que se refieran a esta Licencia y a la ausencia de cualquier garantía; y distribuya una copia de esta Licencia junto con la Biblioteca.

   Puede cobrar un importe por el acto físico de traspasar una copia y puede, a su elección, ofrecer una protección de garantía a cambio de un importe.

2. Puede modificar su copia o copias de la Biblioteca o cualquier parte de ella, formando así un trabajo basado en la Biblioteca, y copiar y distribuir tales modificaciones o trabajo según los términos de la sección 1, siempre que usted también cumpla las siguientes condiciones:

   a. El trabajo modificado debe ser por si mismo una biblioteca.
   b. Los ficheros modificados deberán contener lleven avisos llamativos, declarando que usted cambió los ficheros y la fecha de cualquier cambio.
   c. Debe obtener una licencia para todo el trabajo, sin cargo a terceros, según los términos de esta Licencia.
   d. Si una prestación en la Biblioteca modificada se refiere a una función o a una tabla de datos, que no sea suministrada por un programa de aplicación que usa la prestación como argumento pasado al invocar la prestación, deberá esforzarse de buena fe para asegurarse de que, en caso de que una aplicación no suministre tal función o tabla, la prestación aun funcione y haga que cualquier parte de su prestación siga siendo significativa.

      (Por ejemplo, la función en una biblioteca para calcular raíces cuadradas tiene un propósito bien definido, independientemente de la aplicación. Por tanto, la Subsección 2d exige que cualquier función o tabla suministrada por la

aplicación y usada por esa función sea opcional: si la aplicación no la suministra, la función de raíz cuadrada debe seguir calculando raíces cuadradas).

Estos requisitos se aplican al trabajo modificado como un todo. Si existen secciones identificables de ese trabajo no derivadas de la Biblioteca y que pueden considerarse razonablemente independientes como trabajos separados, entonces esta Licencia y sus términos, no se aplicarán a aquellas secciones cuando usted los distribuya como trabajos separados. Pero cuando usted distribuya estas mismas secciones como parte de un todo, como un trabajo basado en la Biblioteca, la distribución del todo debe estar bajo los términos de esta Licencia, cuyos permisos para otras licencias se extienden a todo el conjunto, y por tanto a todas y cada una de las partes, sin tener en cuenta quien lo escribió.

Así pues, la intención de esta sección no es exigir derechos o discutir los derechos de un trabajo escrito exclusivamente por usted; sino más bien ejercer el derecho a controlar la distribución de trabajos derivados o colectivos basados en la Biblioteca.

Además, la mera agregación de otro trabajo no basado en la Biblioteca a la misma Biblioteca —o a un trabajo basado en la Biblioteca— en un volumen de almacenaje o en un medio de distribución, no coloca este trabajo entre los objetivos de esta Licencia.

3. Usted puede optar por aplicar a una determinada copia de la Biblioteca, los términos de la Licencia Pública General GNU ordinaria en lugar de los de la presente Licencia. Para ello, deberá alterar todas las notificaciones que se refieren a esta Licencia, para que se refieran a la Licencia Pública General GNU ordinaria, versión 2, en lugar de a esta Licencia. (Si ha aparecido una versión más reciente que la versión 2 de la Licencia Pública General GNU ordinaria, entonces, si lo desea, puede especificar esa nueva versión.) No introduzca ningún otro cambio en estas notificaciones.

   Una vez que se haya hecho este cambio en una copia dada, es irreversible para esa copia, de modo que la Licencia Pública General GNU ordinaria se aplica a todas las copias siguientes y a trabajos derivados realizados a partir de esa copia.

   Esta opción es útil cuando usted desea copiar parte del código de la Biblioteca dentro de un programa que no es una biblioteca.

4. Puede copiar y distribuir la Biblioteca —o una porción o derivado de ésta, bajo la Sección 2— en código objeto o forma ejecutable bajo los términos de las Secciones 1 y 2 arriba indicadas, siempre que la acompañe el correspondiente código fuente legible —a máquina— completo, que deberá distribuirse según los términos de las Secciones 1 y 2, en un medio usado habitualmente para el intercambio de software.

   Si la distribución del código objeto se realiza ofreciendo el acceso a su copia desde un lugar designado, entonces la provisión de un acceso equivalente a la copia del

código fuente desde el mismo sitio satisfará los requisitos para la distribución del código fuente, aunque terceras partes no estén obligadas a copiar el código fuente junto con el código objeto.

5. Un programa que no contiene derivado de ninguna porción de la Biblioteca, pero está diseñado para trabajar con la Biblioteca al ser compilado o enlazado con ella, se denomina un «trabajo que utiliza la Biblioteca». Dicho trabajo, por separado, no es un trabajo derivado de la Biblioteca y por tanto cae fuera del ámbito de esta Licencia.

   Sin embargo, al enlazar un «trabajo que utiliza la Biblioteca» con la Biblioteca, se crea un ejecutable que es un derivado de la Biblioteca —ya que contiene porciones de la misma. El ejecutable está por tanto cubierto por esta Licencia. La sección 6 expone los términos para la distribución de tales ejecutables.

   Cuando un «trabajo que utiliza la Biblioteca» utiliza material de un fichero cabecera que forma parte de la Biblioteca, el código objeto del trabajo puede ser un trabajo derivado de la Biblioteca aunque el código fuente no lo sea. Que esto sea cierto es especialmente significativo si el trabajo puede enlazarse sin la Biblioteca, o si el trabajo es en sí mismo una biblioteca. Este matiz no está definido con precisión por la ley.

   Si dicho fichero objeto utiliza solo parámetros numéricos, esquema de estructura de datos, pequeñas macros y pequeñas funciones en línea —diez líneas o menos de longitud—, entonces el uso del fichero objeto no estará restringido, independientemente de que legalmente sea un trabajo derivado. (Ejecutables que contengan este código objeto y porciones de la Biblioteca estarán bajo la Sección 6.

   En caso contrario, si el trabajo es un derivado de la Biblioteca, usted puede distribuir el código objeto del trabajo según los términos de la Sección 6. Cualquier ejecutable que contenga ese trabajo también cae bajo la Sección 6, esté o no enlazado con la Biblioteca.

6. Como excepción a las secciones anteriores, puede también combinar o enlazar un «trabajo que utiliza la Biblioteca» con la Biblioteca para producir un trabajo que contenga porciones de la Biblioteca y distribuir ese trabajo según los términos de su elección, siempre que los términos permitan la modificación del trabajo para el uso propio del cliente y la ingeniería inversa para la depuración de tales modificaciones.

   Debe incluir con cada copia del trabajo una notificación de que la Biblioteca se utiliza en él y de que la Biblioteca y su uso están cubiertos por esta Licencia. Debe suministrar una copia de esta Licencia. Si el trabajo, durante su ejecución, muestra notas de derechos de autor, deberá incluir entre ellas las notas de copyright de la Biblioteca, así como una referencia que dirija al usuario a la copia de esta Licencia. Además, usted debe hacer una de estas cosas:

a. Acompañar el trabajo con el correspondiente código fuente legible —a máquina— completo de la Biblioteca, incluyendo cualquier cambio introducido en el trabajo —distribuido bajo las Secciones 1 y 2—; y, si el trabajo es un ejecutable enlazado con la Biblioteca, con el completo, legible (a maquina) «trabajo que utiliza la Biblioteca», como código objeto y/o código fuente, de forma que el usuario pueda modificar la Biblioteca y reenlazarlo para producir un ejecutable modificado que contenga la Biblioteca modificada. (Se entiende que el usuario que cambia los contenidos de los archivos de definiciones en la Biblioteca no necesariamente será capaz de recompilar la aplicación para usar las definiciones modificadas.)

b. Utilizar un mecanismo de biblioteca compartida adecuado para enlazar con la Biblioteca. Un mecanismo adecuado es aquel que (1) utiliza en tiempo de ejecución una copia de la biblioteca ya presente en el ordenador del usuario, en lugar de copiar funciones de biblioteca dentro del ejecutable y (2) que funciona correctamente con una versión modificada de la biblioteca, si el usuario instala una, mientras que la versión modificada sea de interfaz compatible con la versión con la que se realizó el trabajo.

c. Acompañar el trabajo con una oferta escrita, valida durante al menos tres años, para proporcionar a dicho usuario los materiales especificados en la Subsección 6a, por un precio no superior al coste de la distribución.

d. Si la distribución del trabajo se realiza ofreciendo el acceso a la copia desde un lugar determinado, ofrecer un acceso equivalente para la copia de los materiales especificados anteriormente desde el mismo lugar.

e. Verificar que el usuario ha recibido ya una copia de estos materiales o que usted ya le ha enviado una copia.

Para un ejecutable, la forma requerida del «trabajo que utiliza la Biblioteca» debe incluir todos los programas de datos y utilidades necesitados para reproducir el ejecutable desde el mismo. Sin embargo, como excepción especial, los materiales a distribuir no necesitan incluir nada de lo que se distribuye normalmente —ya sea en forma binaria o fuente— con los componentes principales —compilador, kernel, y demás— del sistema operativo en el cual funciona el ejecutable, a menos que el componente acompañe al ejecutable.

Puede suceder que este requisito contradiga las restricciones de la licencia de otras bibliotecas propietarias que no acompañan normalmente al sistema operativo. Dicha contradicción significa que no puede usar ambas juntas en un ejecutable que usted distribuya.

7. Puede añadir prestaciones de biblioteca, que sean un trabajo basado en la Biblioteca, juntas en una sola biblioteca junto con otras prestaciones de biblioteca no cubiertas por esta Licencia y distribuir dicha biblioteca combinada, con tal de que la distribución separada del trabajo basado en la Biblioteca y de las otras pres-

taciones de biblioteca esté, por lo demás, permitida, y con tal de que usted haga estas dos cosas:

a. Acompañar la biblioteca combinada con una copia del mismo trabajo basado en la Biblioteca, sin combinarlo con otras prestaciones de biblioteca. Esto debe ser distribuido según los términos de las Secciones anteriores.
b. Incluir una notificación en la biblioteca combinada destacando que parte de la misma es un trabajo basado en la Biblioteca, y explicando dónde encontrar las formas sin combinar que acompañan a éste trabajo.

8. No debe copiar, modificar, sublicenciar, enlazar o distribuir la Biblioteca excepto como se estipula expresamente en esta Licencia. Cualquier otro intento de copiar, modificar, sublicenciar, enlazar o distribuir la Biblioteca no será válido y anulará automáticamente sus derechos con relación esta Licencia. Sin embargo, aquellos que hayan recibido copias o derechos según esta Licencia por medio de usted, conservarán sus licencias siempre que cumplan los términos de las mismas.

9. No se exige que acepte esta Licencia, puesto que no la ha firmado. Sin embargo, nada más le autoriza a modificar o distribuir la Biblioteca o los trabajos derivados de la misma. Si usted no acepta esta Licencia, estas acciones están prohibidas por ley. Por lo tanto, al modificar o distribuir la Biblioteca —o cualquier trabajo basado en ella—, usted acepta esta Licencia, y todos sus términos y condiciones para copiar, distribuir o modificar la Biblioteca o los trabajos basados en ella.

10. Cada vez que usted distribuye la Biblioteca —o cualquier trabajo basado en ella—, el receptor recibe automáticamente una licencia del titular original de la Licencia para copiar, distribuir, enlazar o modificar la Biblioteca siempre sujeto a estos términos y condiciones. Usted no debe imponer ninguna restricción posterior sobre el ejercicio de los receptores de los derechos otorgados mencionados aquí. Usted no es responsable de hacer cumplir esta Licencia a terceros.

11. Si, como consecuencia de un juicio o infracción de patente o por cualquier otra razón —no limitada a asuntos de patente— a usted se le imponen condiciones —sea orden judicial, contractual u otras— que contradigan las condiciones de esta Licencia, eso no le dispensa de las condiciones de esta Licencia. Si usted no puede distribuirla de tal forma que satisfaga simultáneamente sus obligaciones con respecto a esta licencia y cualquier otras obligaciones pertinentes, entonces como consecuencia, no debe en absoluto distribuir la Biblioteca. Por ejemplo, si una licencia de patente no permitiera la redistribución libre del copyright de la Biblioteca a todo aquellos que reciben copias directamente o indirectamente a través de usted, entonces deberá abstenerse completamente de distribuir la Biblioteca.

    Si cualquier parte de esta sección se considera nula o inaplicable en cualquier circunstancia particular, se intentará aplicar el grueso de la sección, y en otras circunstancias se intentará aplicar la sección como un todo.

No es el propósito de esta sección inducirle a infringir una demanda de derechos de patente u otros derechos de propiedad o impugnar la validez de tales demandas; esta sección tiene como único propósito proteger la integridad del sistema de distribución de software libre, lo cual se lleva a cabo mediante prácticas de licencia pública. Mucha gente ha hecho generosas contribuciones a una amplia variedad de proyectos de software distribuido mediante este sistema, confiando en la firme aplicación del mismo; es decisión del autor/donante decidir si él o ella desea distribuir software mediante cualquier otro sistema y una licencia no puede imponer esa elección.

Esta sección tiene el propósito de esclarecer a fondo lo que se considera una consecuencia del resto de esta licencia.

12. Si la distribución y/o uso de la Biblioteca está restringida en ciertos países mediante patentes o interfaces con derechos de autor, el propietario de los derechos de autor originales, que puso la Biblioteca bajo esta Licencia, puede añadir una limitación a la distribución geográfica excluyendo a estos países, de forma que esta distribución se autorice solamente en o entre países no excluidos. En tal caso, esta Licencia incorpora la limitación como si estuviera escrita en el cuerpo de esta Licencia.

13. La *Free Software Foundation* puede publicar versiones nuevas y/o revisadas de la Licencia Pública General Menor de cuando en cuando. Tales versiones nuevas serán similares en espíritu a la presente versión, pero pueden diferir en ciertos detalles para abordar nuevos problemas o intereses.

    A cada versión se le asigna un número que la distingue. Si la biblioteca especifica un número de versión de esta Licencia que se aplica a ella misma y a «cualquier versión posterior», usted puede optar por cumplir los términos y condiciones tanto de esa versión como de cualquier versión posterior publicada por la *Free Software Foundation*. Si la Biblioteca no especifica un número de versión de licencia, usted puede elegir cualquier versión publicada por la *Free Software Foundation*.

14. Si desea incorporar partes de la Biblioteca a otros programas libres cuyas condiciones de distribución sean incompatibles con estos, escriba al autor para pedirle permiso. Para el software cuyos copyright pertenece a la *Free Software Foundation*, escriba a la *Free Software Foundation*; a veces, hacemos excepciones. Nuestra decisión se guiará por lo general por los dos objetivos siguientes: preservar el estatus libre de todo lo derivado de nuestro software y promover que el software sea compartido y reutilizado.

SIN GARANTÍA

15. DADO QUE LA BIBLIOTECA ESTÁ LICENCIADA DE FORMA GRATUITA DE CARGO, NO HAY GARANTÍA PARA LA BIBLIOTECA SALVO EN AQUELLO QUE ESTABLECE

LA LEGISLACIÓN. EXCEPTO CUANDO SE ESTABLEZCA DE OTRO MODO POR ESCRITO, LOS TITULARES DEL COPYRIGHT Y/O OTRAS PARTES SUMINISTRAN LA BIBLIOTECA «TAL CUAL» SIN GARANTÍA DE NINGUNA CLASE, YA SEA DE FORMA EXPRESA O IMPLÍCITA, INCLUYENDO, AUNQUE NO LIMITADO POR, LAS GARANTÍAS IMPLÍCITAS DE COMERCIABILIDAD Y CONVENIENCIA PARA UN PROPÓSITO PARTICULAR. TODO RIESGO ASOCIADO A LA CALIDAD Y LA EJECUCIÓN DE LA BIBLIOTECA ES SUYO. SI LA BIBLIOTECA RESULTARA ESTAR DEFECTUOSA, USTED ASUME EL COSTE DE TODO MANTENIMIENTO, REPARACIÓN O CORRECCIÓN NECESARIOS.

16. BAJO NINGÚN CONCEPTO, A MENOS QUE SEA REQUERIDO POR LA LEY APLICABLE O DE ACUERDO A UN ESCRITO, EL TITULAR DEL COPYRIGHT O CUALQUIER OTRA PARTE QUE PUEDA MODIFICAR Y/O REDISTRIBUIR LA BIBLIOTECA COMO SE PERMITE ARRIBA, SERÁ RESPONSABLE POR DAÑOS, INCLUYENDO CUALQUIER DAÑO GENERAL, ESPECIAL, ACCIDENTAL O CONSECUENTE ORIGINADO POR EL USO O INCAPACIDAD DE USO DE LA BIBLIOTECA —INCLUYENDO PERO NO LIMITANDO POR, LAS PÉRDIDAS DE DATOS O LA PRODUCCIÓN DE DATOS INCORRECTOS, O PÉRDIDAS SUFRIDAS POR USTED O TERCEROS, O UN FALLO DE LA BIBLIOTECA PARA FUNCIONAR CON CUALQUIER OTRO SOFTWARE— INCLUSO SI TAL TITULAR U OTRA PARTE HUBIERA SIDO NOTIFICADO DE LA POSIBILIDAD DE TALES DAÑOS.

FIN DE TÉRMINOS Y CONDICIONES

Cómo aplicar estos términos a sus nuevas bibliotecas

Si usted desarrolla una biblioteca nueva, y quiere que sea del mayor uso posible para el publico, le recomendamos que la convierta en software libre para que todo el mundo pueda redistribuirla o combinarla. Usted puede hacer tal cosa autorizando la redistribución según estos términos (o bien según los términos de la Licencia Pública General ordinaria).

Para aplicar estos términos agregue la siguiente nota a la biblioteca. Es más prudente agregar los avisos al comienzo de cada fichero fuente para transmitir de una forma más efectiva la exclusión de garantía; y cada fichero debería tener al menos la línea de derechos de autor y un puntero hacia donde se encuentre la notificación completa.

```
Copyright ©

Esta biblioteca es software libre, puede redistribuirla o
modificarla según los términos de la GNU Licencia Pública
General Menor, publicada por la Free Software Foundation;
```

```
ya sea en su versión 2 o -- a su elección-- en cualquier
versión posterior.

Esta biblioteca se distribuya con la intención de que sea
usada, sin embargo no TIENE GARANTÍA; incluidas la garantías
de comerciabilidad y conveniencia para un propósito
particular. Véase GNU Licencia Pública General Menor para
más detalles.

Deberá recibir una copia de la GNU Licencia Pública General
Menor con esta biblioteca; si no es así, escriba a la Free
Software Foundation, Inc., 59 Temple Place, Suite 330,
Boston, Ma 02111-1307 USA.
```

Añada también información sobre como contactar con usted mediante correo electrónico y postal.

Debería incluir también su empleo —si trabaja como programador— o sus estudios, si tiene alguno, para firmar una «renuncia de copyright» para la biblioteca, si fuese necesario. Aquí tenemos un ejemplo; altere los nombres:

```
Yoyodyne, Inc., por la presente, renuncia a todos sus
derechos de copyright con respecto a la biblioteca «Frob»
--una biblioteca para pellizcar granitos-- escrita por James
Random Hacker.

firma de Ty Coon, 1 April 1990
Ty Coon, Presidente de Vicio
```

¡Eso es todo!

Apéndice C

Licencia de Documentación Libre GNU[1]

Free Document License, GNU-FDL

Versión 1.1, Marzo de 2000

Preámbulo

El propósito de esta licencia es permitir que un manual, libro de texto, u otro documento escrito sea «libre» en el sentido de libertad: asegurar a todo el mundo la libertad efectiva de copiarlo y redistribuirlo, con o sin modificaciones, de manera comercial o no. En segundo término, esta licencia preserva para el autor o para quien lo publique una manera de obtener reconocimiento por su trabajo, al tiempo que no se les hace responsables de las modificaciones realizadas por terceros.

Esta licencia es una especie de «copyleft», lo que significa que los trabajos derivados del documento deben a su vez ser libres en el mismo sentido. Esta complementa a la Licencia Pública General GNU, que es una licencia de copyleft diseñada para el software libre.

Hemos diseñado esta Licencia para usarla en manuales de software libre, ya que el software libre necesita documentación libre: un programa libre debe acompañarse con manuales que ofrezcan la mismas libertades que da el software. Sin embargo esta licencia no se limita a manuales de software; puede utilizarse para cualquier trabajo textual, sin tener en cuenta su temática o si se publica como libro impreso. Recomendamos esta

[1] This is an unofficial translation of the GNU Fre Document License into spanish. It was not published by the Free Software Foundation, and does not legally state the distribution terms that uses the GNU FDL —only the original English text of the GNU FDL does that. [Esta es una traducción no oficial al español de la GNU Free Document License. No ha sido publicada por la Free Software Foundation, y no establece legalmente las condiciones de distribución de manuales y documentación para el software que usa la GNU FDL. Estas condiciones se establecen solamente por el texto original, en inglés, de la GNU FDL.]

licencia principalmente para trabajos cuyo fin sea educativo o de servir como obra de referencia.

Aplicabilidad y definiciones

Esta Licencia se aplica a cualquier manual u otro documento que contenga una nota del propietario de los derechos que indicando que puede distribuirse bajo los términos de la Licencia. El «Documento», en adelante, se refiere a cualquiera de dichos manuales o trabajos. Cualquier miembro del público es un licenciatario y será denominado como «Usted».

Una «versión modificada» del Documento es cualquier trabajo que contenga el Documento o una porción del mismo, ya sea una copia literal o con modificaciones y/o traducciones a otro idioma.

Una «sección secundaria» es un apéndice titulado o una sección preliminar al prólogo del «Documento» que tiene que ver exclusivamente con la relación de quien publica, los autores del «Documento», el tema general del «Documento» —o asuntos relacionados— y cuyo contenido no concierne directamente a este tema general. (Por ejemplo, si el «Documento» es en parte un texto de matemáticas, una «Sección Secundaria» puede no explicar matemáticas.) La relación puede ser un asunto sobre la relación histórica o el posicionamiento legal, comercial, filosófico, ético o político con respecto del tema o la materia del texto.

Las «secciones Invariables» son ciertas secciones secundarias cuyos títulos son denominados como secciones invariables, en la nota que indica que el documento es liberado bajo esta licencia.

Los «textos de cubierta» son ciertos pasajes breves que se enumeran, como textos de portada o textos de contra portada, en la nota que indica que el documento es liberado bajo esta Licencia.

Una copia «transparente» del Documento es una copia para lectura en máquina, representada en un formato cuya especificación está a disposición del público general, cuyo contenido puede ser leído y editado directamente con editores de texto genéricos o —para imágenes compuestas por píxeles— de programas genéricos de dibujo o —para dibujos— algún editor gráfico disponible, y que sea adecuado para exportar a formateadores de texto o para la traducción automática a una variedad de formatos adecuados para utilizarlos con formateadores de texto. Una copia en un formato de archivo no transparente, diseñado para impedir o dificultar subsecuentes modificaciones posteriores por parte de los lectores no es transparente. Una copia que no es «transparente» es llamada «opaca».

Como ejemplos de formatos adecuados para copias transparentes están el ASCII plano sin formato, formato de Texinfo, formato de LaTeX, SGML o XML usando un DTD disponible y HTML simple que obedece a estándares, diseñado para modificaciones humanas. Los formatos opacos incluyen PostScript, PDF, formatos propietarios que pueden ser leídos y editados únicamente en procesadores de textos propietarios, SGML o XML para los cuáles los DTD y/o herramientas de procesamiento no están

generalmente disponibles, y el HTML generado por máquinas producto de algún procesador de textos sólo con fines de salida.

La «portada» en un libro impreso es la portada misma, más las páginas siguientes necesarias para mantener la legibilidad del material, que esta Licencia requiere que aparezca en la portada. Para trabajos en formatos que no tienen portada como tal, la «portada» es el texto más próximo al título del trabajo, precediendo el comienzo del cuerpo del trabajo.

Copia literal

Puede copiar y distribuir el Documento en cualquier medio, sea en forma comercial o no, siempre y cuando esta Licencia, las notas sobre copyright y la nota de licencia, que indica que esta Licencia se aplica al Documento, se reproduzca en todas las copias y que usted no añada ninguna condición aparte de las expuestas en esta Licencia. No puede utilizar medidas técnicas para obstruir o controlar la lectura o copia posterior de las copias que usted haga o distribuya. Sin embargo, usted puede aceptar una compensación monetaria a cambio de las copias. Si distribuye un número suficientemente grande de copias también deberá cumplir las condiciones de la sección 3.

También puede prestar copias, bajo las mismas condiciones establecidas anteriormente y puede exhibir copias públicamente.

Copia en cantidades masivas

Si publica más de cien copias impresas del Documento y la nota de Licencia del Documento exige textos de cubierta, debe incluir las copias con cubiertas que muestren de forma clara y legible, todos los textos de cubierta: textos frontales en la cubierta frontal y textos posteriores de cubierta en la cubierta posterior. Ambas cubiertas deben identificarle a usted, de forma clara y legible como responsable de la publicación de tales copias. La cubierta frontal debe mostrar el título completo siendo todas las palabras igualmente prominentes y visibles. Además, puede añadir otro material en la cubierta. Las copias con cambios limitados en las cubiertas, siempre que preserven el título del Documento y satisfagan estas condiciones, puede considerarse como copias literales.

Si los textos requeridos para la cubierta son muy voluminosos para que se ajusten de forma legible, deberá colocar los primeros —tantos como sea razonable— en la cubierta real, e introducir el resto en las páginas adyacentes.

Si publica o distribuye más de cien copias opacas del Documento, deberá incluir una copia transparente que pueda ser leída por una máquina con cada copia opaca, o entregar con cada copia opaca una dirección pública en red que contenga una copia completa transparente del Documento, sin material adicional, a la cual el público en general pueda acceder y bajar anónimamente sin cargo, usando protocolos de estándar público. Si recurre a esta última opción, deberá tomar las medidas necesarias, cuando comience la distribución de las copias opacas, para asegurarse de que esta copia transparente permanecerá accesible en el sitio por lo menos un año después de su

última distribución de copias opacas —ya sea directamente o a través de sus agentes o distribuidores— de esa misma edición pública.

Se solicita, aunque no es requisito, que se ponga en contacto con los autores del Documento antes de redistribuir cualquier un número de copias, para darles la oportunidad de que le proporcionen una versión del Documento.

Modificaciones

Puede copiar y distribuir una versión modificada del Documento según las condiciones de las anteriores secciones 2 y 3, siempre que usted libere la versión modificada con esta misma Licencia, de este modo, la versión modificada desempeña el papel del Documento, y por lo tanto se autoriza la distribución y la modificación de la versión modificada a quienquiera que posea una copia de éste. Además, deberá hacer lo siguiente en la versión modificada:

a. Usar en la Portada —y en las cubiertas, si las hubiera— un título distinto al del Documento y de las versiones anteriores —que deberían, si las hubiera, enumerarse en la sección de Historia del Documento. Puede utilizar el mismo título de las versiones anteriores del original siempre que quien publicara la primera versión le de su consentimiento.

b. Enumerar en la Portada, como autores, a una o más personas o entidades responsables de la autoría o de las modificaciones de la versión modificada, junto con por lo menos cinco de los autores principales del Documento —todos sus autores principales, si hay menos de cinco.

c. Incluir en la portada el nombre de quién publique la versión modificada.

d. Preservar todas las notas de copyright del Documento.

e. Añadir una nota de copyright apropiada para sus modificaciones adyacentes a las otras notas de copyright.

f. Incluir, inmediatamente después de la nota de copyright, una nota de licencia autorizando el uso de la versión modificada según los términos de esta Licencia, de la forma descrita en la addenda.

g. Preservar en esa nota de licencia la lista completa de secciones invariables y en los textos de las cubiertas que sean requeridos según se especifique en la nota de Licencia del Documento

h. Incluir una copia sin modificación de esta Licencia.

i. Preservar la sección llamada «historia» y su título, y añadir a ésta una sección que establezca al menos el título, el año, los nuevos autores y quién publicó la versión modificada tal y como se especifica en la portada. Si no hay una sección titulada

«historia» en el Documento, se creará una estableciendo el título, el año, los autores y quien publicó el Documento como se especifica en la portada, añadiendo además un artículo que describa la versión modificada como se estableció en el punto anterior.

j. Preservar la localización en red, si la hubiera, especificada en la Documentación para acceder públicamente a una copia transparente del Documento, al igual que otras direcciones de red proporcionadas en el Documento para versiones anteriores en las cuales estuviese basado. Estas pueden ubicarse en la sección «Historia». Se puede omitir la ubicación en red para un trabajo publicado por lo menos 4 años antes que el Documento, o si quien publica originalmente la versión da su consentimiento explícitamente.

k. En cualquier sección titulada «agradecimientos» o «dedicatorias», se preservará el título de la sección, así como la intención y el tono de los agradecimientos y/o dedicatorias de cada contribuyente.

l. Preservar todas las secciones invariables del Documento, sin alterar su contenido ni sus títulos. Los números de sección o el equivalente no se consideran parte de los títulos de la sección.

m. Suprimir cualquier sección titulada «aprobaciones». Tales secciones no pueden estar incluidas en las versiones modificadas.

n. No retitular ninguna sección existente como «aprobaciones», de modo que pueda entrar en conflicto con el título de alguna sección invariante.

Si la versión modificada incluye secciones o apéndices nuevos o preliminares al prólogo calificados como secciones secundarias que contienen material no copiado del Documento, opcionalmente, puede designarse algunas o todas estas secciones como invariables. Para ello, añada sus títulos a la lista de secciones invariables en la nota de licencia de la versión modificada. Tales títulos deben ser distintos de cualquier otro título de sección.

Puede añadir una sección titulada «aprobaciones», siempre que contenga únicamente las aprobaciones de su versión modificada por diversas fuentes —por ejemplo, observaciones de peritos o notas exponiendo que el texto ha sido aprobado por una organización como estándar.

Puede añadir un pasaje de hasta cinco palabras como texto de cubierta frontal, y un pasaje de hasta 25 palabras como texto de cubierta posterior, al final de la lista de textos de cubierta en la versión modificada. Solamente puede añadir un pasaje de texto de cubierta frontal y un texto de cubierta posterior —ya sea a manera de arreglos hechos por una entidad. Si el Documento ya incluye un texto de cubierta para la misma cubierta, previamente añadido por usted o por la misma entidad, en nombre de la cual está actuando, no puede añadir más; pero puede reemplazar el anterior, con autorización expresa de quien publicó anteriormente la cubierta.

El(los) autor(es) y quien(es) publica(n) el Documento no autorizan con esta Licencia permiso alguno que sus nombres sean utilizados en publicidad o para asegurar o sugerir la aprobación de cualquier Versión Modificada.

Combinar documentos

Puede combinar el Documento con otros documentos liberados bajo esta Licencia, según los términos definidos en la sección 4 anterior para versiones modificadas, siempre que incluya en la combinación todas las secciones invariables de los documentos originales, sin modificar y enumeradas como secciones invariables del trabajo combinado en su nota de licencia.

El trabajo combinado debe contener solamente una copia de esta Licencia, y las múltiples secciones invariables idénticas pueden ser reemplazadas por una sola copia. Si hay múltiples secciones invariables con el mismo nombre pero con contenidos diferentes, haga que el título de cada una de estas secciones sea único añadiéndole al final de este, en paréntesis, el nombre del autor o de quien publicó originalmente esa sección, si es conocido, o si no, un número único. Haga lo mismo con los títulos de sección en la lista de Secciones Invariables en la nota de licencia del trabajo combinado.

Deberá combinar cualquier sección titulada «historia» de los diversos documentos originales, formando una sola sección titulada «historia»; de la misma forma combinará cualquier sección titulada «agradecimientos» y cualquier sección titulada «dedicatorias». Deberá borrar todas las secciones tituladas «aprobaciones».

Colecciones de documentos

Puede hacer una colección consistente en el Documento y en otros documentos liberados bajo esta Licencia y reemplazar las copias individuales de esta Licencia en los diversos documentos con una sola copia incluida en la colección, siempre que siga las reglas de esta Licencia para una copia literal de cada uno de los documentos en cualquiera de todos los aspectos.

Puede extraer un solo documento de tales colecciones y distribuirlo individualmente bajo esta Licencia, siempre que inserte una copia de esta Licencia en el documento extraído y cumpla esta Licencia en todos los demás aspectos concernientes a la copia literal de tal documento.

Combinación con trabajos independientes

Una recopilación del Documento o de sus derivados con otros documentos o trabajos separados o independientes, en cualquier tipo de distribución o medio de almacenamiento, no como un todo, se considerará como una versión modificada del Documento, teniendo en cuenta que ninguna compilación de copyright sea clamada por la recopilación. Tal recopilación es llamada un «agregado» y esta Licencia no se aplica a los otros

trabajos auto-contenidos y por lo tanto compilados con el Documento, si no se trata de trabajos derivados del Documento.

En caso de que la sección 3 del texto de la cubierta fuera aplicable a estas copias, si el Documento es inferior a un cuarto del agregado entero, los textos de la cubierta del Documento pueden ser colocados en cubiertas que enmarquen solamente el Documento en el agregado. De otra forma deben aparecer en cubiertas enmarcando todo el agregado.

Traducción

La Traducción se considera un tipo de modificación. De este modo, puede distribuir traducciones del Documento bajo los términos de la sección 4. La sustitución de las secciones invariables por traducciones requerirá la autorización de los titulares del copyright, pero puede incluir traducciones de algunas o todas las Secciones Invariables además de las versiones originales de las Secciones Invariables. Puede incluirse una traducción de esta Licencia siempre que incluya también la versión inglesa. En caso de que exista un desacuerdo entre la traducción y la versión original en inglés de esta Licencia, prevalecerá la versión original en inglés.

Nulidad

No se puede copiar, modificar, sublicenciar o distribuir el Documento excepto en los casos expresamente autorizados en esta Licencia. Cualquier otro intento de copia, modificación, sublicenciamiento o distribución del Documento será nulo y sus derechos serán automáticamente anulados bajo esa licencia. De todas maneras, los terceros que hayan recibido copias o derechos, de su parte, bajo esta Licencia no tendrán por anuladas sus licencias siempre que tales personas o entidades se encuentren en total conformidad con la licencia original.

Futuras revisiones de esta licencia

La *Free Software Foundation* puede publicar nuevas versiones revisadas de la Licencia de Documentación Libre GNU de cuando en cuando. Estas nuevas versiones serán similares en espíritu a la presente versión, pero pueden diferir en algunos aspectos con el fin de solucionar algunos problemas o intereses. Véase http://www.gnu.org/copyleft/.

Cada versión de la Licencia tiene un número que la distingue de las demás. Si el Documento especifica que una versión numerada de esta licencia o «cualquier versión posterior» se aplica al mismo, tendrá la opción de cumplir los términos y condiciones de la versión especificada o de cualquier versión posterior que haya sido publicada —no como borrador— por la *Free Software Foundation*. Si el Documento no especifica un

número de versión para esta Licencia, puede escoger cualquier versión que haya sido publicada —no como borrador— por la *Free Software Foundation*.

Addenda

Para utilizar esta licencia en un documento que usted haya escrito, incluya una copia de la Licencia en el documento y añada la siguiente nota sobre el copyright y la licencia justo después del título de la página:

```
Copyright © Año Su Nombre.

Permiso para copiar, distribuir y/o modificar este documento
según los términos de la Licencia de Documentación Libre
GNU, Versión 1.1 o cualquier otra versión posterior
publicada por la Free Software Foundation; con las Secciones
Invariables ENUMERE SUS TÍTULOS, siendo INTRODUZCA el texto
de la cubierta frontal y siendo INTRODUZCA el texto de la
cubierta posterior. Se incluye una copia de la licencia en
la sección titulada «Licencia de Documentación Libre GNU».
```

Si no tiene Secciones Invariables, escriba «sin secciones invariables» en lugar de decir cuáles son invariables. Si no tiene texto de cubierta frontal, escriba «sin texto de cubierta frontal». Haciendo lo propio con la Cubierta Posterior.

Si su documento contiene ejemplos de código de programa no triviales, recomendamos liberar también estos ejemplos con la elección de una licencia de software libre, como la Licencia de Público General GNU, para permitir su uso en software libre.